Bull. Nov. Comp. Center, Comp. Science, 27 (2008), 1–39

Cellular Automata composition techniques for spatial dynamics simulation

Olga Bandman

Abstract. A Cellular Automaton (CA) is nowadays an object of ever growing interest as a mathematical model for spatial dynamics simulation. Due to the CA ability to simulate nonlinear and discontinuous processes, it is expected to become a complement to partial differential equations. Particularly, the CA may be helpful when there is no other mathematical model of a phenomenon to be simulated. The fact is, there is no formal procedure to construct a CA-model according to a given qualitative or a quantitative specification of a space–time process. But, there exists a relatively large bank of CA that may be used for constructing the new complex CA models out of several known simple ones. In order to exploit this possibility CA composition methods are needed. The main purpose of this paper is to present a theoretical foundation and its basis on CA composition techniques in a generalized and systematic form. To capture all features of a great diversity of CA-models, a more general formalism for CA-algorithms representation, namely, Parallel Substitution Algorithm is chosen as a mathematical tool. The paper combines the results about the subject under consideration that are scattered in publications, most of them being original.

1. Introduction

A Cellular Automaton (CA) is nowadays an object of ever growing interest as a mathematical model for the spatial dynamics simulation. Due to its ability to simulate nonlinear and discontinuous processes, CA is expected [1, 2] to become a complement to partial differential equations (PDE). Particularly, CA may be helpful when there is no other mathematical model of a phenomenon to be simulated. By now, a great variety of CA are known, whose evolution simulates certain kinds of spatial dynamics. All of them are descendants of a classical von-Neumann's CA, which has a Boolean alphabet, deterministic single-cell updating transition functions, and a synchronous mode of operation. Such CA are capable of simulating a number of processes in physics, chemistry, biology and sociology. The most known are CA-models of physical processes, such as diffusion [1, 3], wave propagation [4], phase transition [2, 5], spatial self-organization [6], etc. More complicated CA called the Gas–Lattice models [7] are used in hydrodynamics, some of them [8, 9] dealing with a real alphabet. Among the above CA-models, there are those, which may be described in terms of PDE. Such are the following: diffusion, liquid flow, solitons. The first two are proved to correspond strictly to their PDE counterparts [10, 11]. As for soliton [4],

its CA-model is a bright manifestation of an essential difference in complexity between third order nonlinear PDE and a simple Boolean function that simulate the same phenomenon.

In chemistry and microelectronics [14], asynchronous probabilistic CA with multi-adjustable cells (in chemistry being referred to as Monte Carlo methods), are intensively used for studying a surface reaction on catalysts [12, 13] and processes of the epitaxial growth of crystals [14]. The processes are simulated by mimicking real movements and interactions of atoms and molecules. This type of processes have no mathematical description in terms of PDE, because continuous functions cannot capture the behavior of discrete particles.

Biology and medicine present a wide range of processes to be simulated by CA, genetics [15], myxobacteria swarming [16], the growth of tumors [17] being the examples. In solving ecological problems, CA are used more and more frequently to simulate the propagation of diseases [18], the growth of tumors, etc. Recently, CA-models have aroused considerable interest in simulating the crowds behavior [19, 20]. Moreover, the CA simulation has now gone beyond the scope of scientific research, being used, for example, to simulate the process of cement hardening [21].

The diversity of processes being simulated by CA caused the necessity to extend the cellular automaton concept by allowing it to have any kind of an alphabet (Boolean, integer, real, symbolic), any kind of transition functions (deterministic, probabilistic), any mode of functioning (synchronous, asynchronous). Although the imperative properties of CA still remain. They are as follows:

- CA consists of many identical simple processing units (cells);
- Interactions between cells are constrained by a small (relatively to the total amount of cells) neighborhood.

Such an extended concept of CA is sometimes referred to as *fine-grained algorithms* [22] or *complex systems* [23]. Nevertheless, hereafter the term CA or CA-model is used as the most habitual one. In Figure 1, CA-model types are collected and allocated according to their properties. It is shown that Cellular Neural Networks (CNN) [24] and an explicit form of discrete representation of Partial Differential Equations (PDE) are also regarded as special cases of CA-models.

Unfortunately, there is no formal procedure to construct a CA-model according to a given qualitative or quantitative specification of a space–time process. All known models are the result of a trial and error work based on a high level of experience in CA modeling, and sophisticated understanding of a phenomenon to be simulated. But now, having a relatively large bank of CA-models as well as powerful computing systems at hand, the problem arises how to construct CA-models of complicated real life processes, which

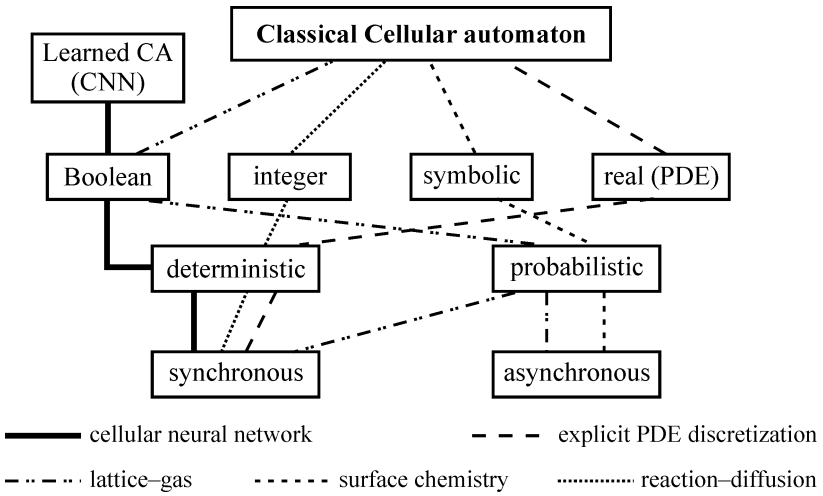


Figure 1. Properties of CA simulation models: lines connecting the rectangles show the sets of properties, characterizing certain types of CA-models

may be thought of as a set of interacting simple ones, for whom CA-models are known. Hence, methods and tools are needed to organize the interaction of CA in such a way that the evolution of a composed CA represents the required spatial process. The problem is similar to that in mathematical physics where a PDE is represented as the sum of differential operators of certain degrees, each term having its own physical meaning.

Complications in the CA composition are associated with the Boolean alphabet, because the overall impact of a Boolean CA-component may not be obtained by means of conventional arithmetics. Even more difficult is to compose CA-models having different alphabets and/or different modes of operation which is the case in reaction–diffusion and prey–predatory processes, when diffusion is given as a Boolean CA, and reaction — as a real function. For example, the snowflakes formation is usually simulated by a Boolean CA, while if it proceeds in an active medium, a chemical component should be added, which may be given as a nonlinear real function. The first prototype of such a composition is proposed in [26] for combining a nonlinear reaction function with a Boolean diffusion, and a more general probabilistic variant is given in [27].

From the above it follows that the CA-composition methods should be capable of performing algebraic operations on CA-configurations with all kinds of admissible numerical alphabets: Boolean, real and integer. This means that some kind of equivalent transformation of CA should be introduced to make CA-models compatible with different alphabets. Based on such a transformation, a special algebra on CA configurations is to be constructed, which allows us to combine the functioning of several CA-models in a single complex process.

A fast increase of the variety of CA-models and the growing necessity of simulating complicated processes require a general formal approach to the CA composition, which is to be valid for any type of CA and any type of their interaction. It is precisely the object of this paper, which aims at presenting a theoretical foundation, and based on it, the CA composition techniques in a generalized and systematic form. To capture all features of an essential diversity of CA-models, a more general formalism for the CA-algorithms representation, namely, Parallel Substitution Algorithm (PSA) [25], is chosen as a mathematical tool.

The paper combines the results on the subject that are scattered about previous papers. It consists of the following sections. In the next section, main concepts and formal definitions are given and operations on cellular arrays are defined. Third section presents a sequential composition of CA-models. In the fourth section a parallel and a mixed composition methods are given. The fifth section is concerned in computational properties of composed CA, namely, accuracy, stability and complexity. All composition methods are illustrated by the original simulation results.

2. Main concepts and formal problem statement

For simulation of the spatial dynamics, an extended concept of CA-model is considered, whose expressive power is sufficient for simulating natural phenomena of several kinds. The concept is based on the PSA formalism [25], which, though intended for the parallel hardware design, seems to be the most suitable for modeling complex processes. Moreover, due to its flexibility, the PSA allows a strict formulation of the main principles of CA composition: 1) conservation of behavioral correctness, and 2) alphabets compatibility of CA involved in the composition.

2.1. Formal definition of a CA-model

Simulation of a natural phenomenon comprises the determination of a suitable mathematical model and computation of a function of time and space. If a CA is chosen as a mathematical model, then time is a discrete sequence $0, 1, \ldots, t, t+1, \ldots$ of natural numbers, space is a discrete set referred to as a *naming set*, function values are from an appropriate *alphabet*.

A finite naming set $M = \{m_k : k = 0, \ldots, |M|\}$ is further taken for the space. Its elements $m_k \in M$ in the simulation tasks are usually represented by the integer vectors of coordinates of a Cartesian space of a finite size. For example, in the 2D case, $M = \{(i, j) : i = 0, 1, \ldots, I,\ j = 0, 1, \ldots, J\}$. The notation m is used instead of (i, j) for making the general expressions shorter and for indicating that they are valid for any other kind of discrete space points.

No constraint is imposed on the alphabet A. The following cases are further used: $A_S = \{a, b, \ldots, n\}$ — a finite set of symbols, $A_B = \{0, 1\}$ — a Boolean alphabet, $A_R = [0, 1]$ — a set of real numbers in a closed interval. Symbols from the second part of the Latin alphabet $\{v, u, x, y, \ldots, z\}$ are used to denote the variables defined on A. Appealing to the above extended concept, the alphabet is dictated by the aim of the study—to combine several CA of different types into a single one for simulating complex phenomena. A pair (a, m) is called a *cell*, a being a cell-state and m being a cell-name. To indicate the state of a cell named by m both notations $u(m)$ and u_m are further used.

The set of cells

$$\Omega = \{(u, m) : \ u \in A, \ m \in M\}, \tag{1}$$

such that there are no cells with identical names is called a *cellular array*, or, sometimes, a *global configuration* of a CA.

Over the naming set M, a mapping $\phi : M \to M$ is defined, referred to as a *naming function*. It determines a neighboring cell location $\phi(m)$ of any cell named m. In the naming set $M = \{(i, j)\}$, naming functions are usually given in the form of the shifts $\phi_k = (i + a, j + b)$, a, b being integers. The set of naming functions determines a *template*

$$T(m) = \{\phi_0(m), \phi_1(m), \ldots, \phi_n(m)\}, \tag{2}$$

which associates a number of cell names to each name $m \in M$. The cell named as $\phi_0(m)$ is called an *active cell* of a template, where $n \ll |M|$, and $\phi_0(m) = m$ by condition.

A subset of cells

$$S(m) = \{(u_0, m), (u_1, \phi_1(m)), \ldots, (u_n, \phi_n(m))\}, \tag{3}$$

with the names from $T(m)$ is called a *local configuration*, with $T(m)$ being its *underlying template*. The set

$$U_S(m) = \{u_0, u_1, \ldots, u_n\}$$

forms a set of local configuration state variables.

A cell (u_k, m) changes its state u_k to the next–state u'_k under the action of a *local operator*, which is expressed in the form of *substitution* [25] as follows

$$\theta(m) : S(m) * S''(m) \to S'(m), \tag{4}$$

where

$$\begin{aligned} S(m) &= \{(v_0, m), (v_1, \phi_1(m)), \ldots, (v_n, \phi_n(m))\}, \\ S'(m) &= \{(u'_0, m), (u'_1, \phi_1(m)), \ldots, (u'_n, \phi_n(m))\}, \\ S''(m) &= \{(v_{n+1}, \phi_{n+1}(m), \ldots, (v_{n+h}, \phi_{n+h}(m))\}, \end{aligned} \tag{5}$$

where $S(m)$, $S'(m)$, and $S''(m)$ are local configurations, the first two having the same underlying template, and the third one comprises h additional cells, which together with $S(m)$ conditions the next state values u'_k.

The next-states u'_k, $k = 0, 1, \ldots, n$, of the cells from S' are computed as values of the *transition functions* f_k, of the states of cells from $S(m) \cup S''(m)$, i.e.,

$$u'_k = f_k(v_0, \ldots, v_n, \ldots, v_{n+h}), \quad \forall\, k = 0, 1, \ldots n. \tag{6}$$

A union of the left-hand side local configurations in (4) $S(m) \cup S''(m)$ is called a *cell-neighborhood*, where S'' is a *context*, $S(m)$ is a *base* of $\theta(m)$. The right-hand side $S'(m)$ is the *next-state base* of the local operator. The underlying templates $T(m)$, $T'(m)$, and $T''(m)$ of the local configuration in (5) are in the following relation:

$$T'(m) = T(m), \qquad T(m) \cap T''(m) = \emptyset, \tag{7}$$

$T(m)$ being referred to as the *basic template* of θ.

A local operator $\theta(m)$ is said to be applicable to a cell named $m \in M$ if $S(m) \cup S''(m) \subseteq \Omega$. Otherwise, it is not applicable. Application of $\theta(m)$ to a certain cell (v, m) (*a single-shot application*) means the following actions. For all $k = 0, \ldots, n$

1) the next-states u'_k are computed according to (6),
2) the cells $(v_k, \phi_k(m)) \in S(m)$ are updated by replacing the cell states u_k by u'_k.

The cells $(v_{n+l}, \phi_{n+l}(m))$, $l = 0, \ldots, h$, from the context remain unchanged. They play the role of an application condition, the states being used as variables in the transition functions (Figure 2).

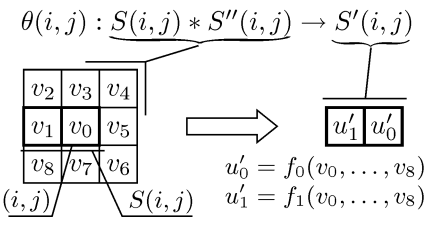


Figure 2. Graphical representation of a local operator

A subset $\hat{M} \subseteq M$, referred to as an *active naming set* is defined, such that it comprises the names of active cells, i.e., the cells to which the local operator is applied to make a cellular array transit to the next state. Application of θ to all active cells $m \in \hat{M}$ comprises an *iteration* performing a *global transition*:

$$\Phi(\hat{M}) : \Omega(t) \to \Omega(t+1). \tag{8}$$

A sequence of iterations results

$$\Sigma(\Omega) = (\Omega, \Omega(1), \ldots, \Omega(t), \Omega(t+1), \ldots, \Omega(\hat{t})) \tag{9}$$

is called *CA-evolution*.

The CA-evolution is a result of a simulation task, representing the process under simulation. If the process converges to a stable global state, then the CA–evolution has termination, i.e., there exists such $t = \hat{t}$, that

$$\Omega(\hat{t}) = \Omega(\hat{t}+1) = \Omega(\hat{t}+2) = \ldots = \Omega(\hat{t}+\xi), \tag{10}$$

where ξ is a priori given number. If this not so, then the evolution is infinite, i.e., exhibits an oscillatory or a chaotic behavior [2].

There are different modes of the local operator application ordering in space and time to perform a global transition from $\Omega(t)$ to $\Omega(t+1)$. The following three are the most important ones.

Synchronous mode provides for transition functions (6) to be computed using the current state values of their variables, i.e.,

$$S(m) \cup S''(m) \subset \Omega(t). \tag{11}$$

The transition to the next cell-state values occurs after all the transition functions in cells from $S(m)$ for all $m \in \hat{M}$ are computed. Theoretically, it may be done in all cells simultaneously or in any order, which manifests the *cellular parallelism*. In fact, when a conventional sequential computer is used, such a cellular parallelism is imitated by delaying the cell updating until all the next states are obtained. So, the cellular parallelism is a *virtual parallelism*, which cannot be for the benefit when a CA-model is run on conventional computers.

Asynchronous mode of operation offers no simultaneous operation (neither real nor virtual). The intrinsic parallelism of the CA is exhibited by an arbitrary order of cells to be chosen for application of $\theta(m)$, the updating of cell states of $S'(m)$ being done immediately after $\theta(m)$ is applied. The time of such an application is referred to further as *time-step* τ. So, each global transition $\Omega(t) \rightarrow \Omega(t+1)$ consists of $|\hat{M}|$ sequential time steps, forming a sequence of cellular arrays

$$\gamma_\alpha(\Omega(t)) = \Omega(t), \Omega(t+\tau), \ldots, \Omega(t+|\hat{M}|\tau), \tag{12}$$

which is referred to as *global state transition sequence*. An important property of the asynchronous mode of operation is that the state values used by transition functions (4) may belong both to $\Omega(t)$ and to $\Omega(t+1)$, i.e.,

$$S(m) \cup S''(m) \subset (\Omega(t) \cup \Omega(t+1). \tag{13}$$

It is the reason why two CA with equal $\langle A, M, \hat{M}, \theta \rangle$ starting from the same Ω may have quite different evolutions when operating in different mode. Although, some exotic "very good" CA are known, whose evolutions and attractors are invariant whatever mode of operation is used [25].

Multi-stage synchronous mode is also frequently used. It is a mixed mode. A multi-stage synchronous CA may be regarded both as a synchronized asynchronous CA, and as an asynchronized synchronous one. The whole cellular array of the CA is partitioned into non-intersecting *blocks* each containing b cells. The block partition induces another naming set partition, called *stage partition* $\{\hat{M}_1, \ldots, \hat{M}_b\}$, whose subsets contain representative names of all blocks, so that $\hat{M} = \hat{M}_1 \cup \ldots \cup \hat{M}_b$. Respectively, the iteration is divided into b stages. At each kth stage, the local operator is applied to the cells of $\hat{M}_k$ synchronously, the stages being processed in an asynchronous manner. Naturally, the cellular parallelism here is limited by the subset cardinality.

No matter what is the mode of operation, a *global operator* is the result of application of $\theta(m)$ to all cells $m \in \hat{M}$.

From the above it follows that a *CA-model*, denoted as $\aleph$ is identified by the five notions:

$$\aleph = \langle A, M, \hat{M}, \theta, \rho \rangle,$$

where ρ indicates to the mode of operation, $\rho = \sigma$ stands for the synchronous mode, $\rho = \beta$ stands for the multistage synchronous mode, and $\rho = \alpha$ — for the asynchronous mode of the local operator application. When the indication of the operation mode is essential, the corresponding symbol is placed as an subindex, e.g. $\aleph_\alpha$ denotes an asynchronous CA.

2.2. Correctness of the local operator interaction

A CA-model $\aleph = \langle A, M, \hat{M}, \theta, \rho \rangle$ is called to be correct (in the computational sense) if its operation satisfies the following correctness conditions.

Non-contradictoriness. *At any moment of time, a cell is allowed to be updated by only one local operator application.* Non-contradictoriness provides the absence of conflicts, i.e., such a situation when a local operator being applied to the cells m and $\phi_k(m)$ simultaneously is attempting to update one and the same cell by writing into it different state values. Formally, the non-contradictoriness sufficient condition is formulated as follows [25]: the simultaneous application of a local operator to m_k and m_l is allowed only if

$$T'(m_k) \cap T'(m_l) = \emptyset \qquad \forall (m_k, m_l) \in M. \tag{14}$$

It is quite clear that the non-contradictoriness condition is always satisfied for classical synchronous CA whose local operator has a single-cell base, i.e. $|S'(m)| = 1$. It is not so if $|S'(m)| > 1$, because the local operator has to change several cells simultaneously. For example, a conflict occurs when a surface chemical reaction is simulated, where there are pairs of molecules occurring in contact (in the adjacent cells) which produce pair of other molecules. In the CA-model this corresponds to changing the states in both cells simultaneously, which leads to a conflict if the synchronous mode is used (Figure 3).

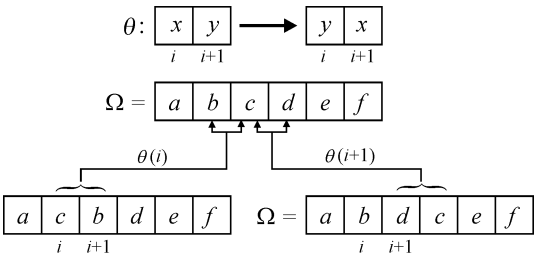


Figure 3. A graphical representation of the contradictoriness property. Simultaneous application of $\theta(i)$ and $\theta(i+1)$ have different results in the cells named $i, i+1, i+2$, engendering a conflict

To avoid the above conflict situation, one has to sacrifice a bit of cellular parallelism to non-contradictoriness. This may be done either by constructing an asynchronous CA, whose evolution simulates the process, or by replacing the synchronous CA $\aleph_\sigma = \langle A, M, \hat{M}, \theta, \sigma \rangle$ by an equivalent multi-stage CA $\aleph_\beta = \langle A, M, \hat{M}_1, \ldots, \hat{M}_b, \theta, \beta \rangle$. Such a sequalization is done according to the following algorithm.

1. The naming set M is partitioned into $|M|/b$ *blocks*, a block being defined by the underlying template

$$B(m) = \{\psi_0(m), \psi_1(m), \ldots, \psi_l(m), \ldots, \psi_b(m)\}$$

in such a way, that

$$\begin{aligned} &B(m_j) \supseteq T'(m_j), && \forall j = 1, \ldots, |M|/b; \\ &\bigcup_{j=1}^{|M|/b} B(m_j) = M, && \forall j = 1, \ldots, |M|/b; \\ &B(m_h) \cap B(m_g) = \emptyset, && \forall m_h, m_g \in \hat{M}_k, \quad \forall k = 1, \ldots, b, \end{aligned} \tag{15}$$

where $T'(m)$ is the basic template in θ.

2. On the active naming set $\hat{M}$, a stage partition $\{\hat{M}_1, \ldots, \hat{M}_k, \ldots, \hat{M}_b\}$ is defined, i.e.,

$$\hat{M}_k = \{\psi_k(m_j) : \ k = 1, \ldots, b; \ j = 1, \ldots, |M|/b\}, \tag{16}$$

$m_j = \psi_0(m_j)$ being an active cell of a block $B(m_j) \in M$.

3. Each iteration $\Omega(t) \to \Omega(t+1)$ is divided into b sequential stages $(t_1, t_2, \ldots, t_b)$, $t_b = t + 1$, the resulting arrays forming a sequence:

$$\gamma_\beta(t) = (\Omega(t), \ldots, \Omega(t + t_k), \Omega(t + t_{k+1}), \ldots, \Omega(t+1), \quad t_k = \frac{\tau k}{b}, \tag{17}$$

that are referred to as *stage transition sequence*. At the k-th stage, $k = 1, \ldots, b$, $\theta(m)$ is applied synchronously to all cells from M'_k.

4. The subsets $\hat{M}_k$, $k = 1, \ldots, b$, are sequentially processed in arbitrary order, hence, the total number of possible stage transition sequences is $|\{\gamma_\beta\}| = b\,!$.

As for asynchronous CA, they always satisfy non-contradictoriness conditions, because at each step only one application of $\theta(m)$ is allowed.

Fairness. *At each iteration, $\theta(m)$ should be applied to all cells $m \in \hat{M}$, to any cell $m \in \hat{M}$ being applied only once.* The fairness ensures that all cells have equal rights to participate in the CA operation process, therefore, it is sometimes referred to as equality of cells [29]. Synchronous classical CA satisfy this property according to the definition of synchronicity. When the multi-stage synchronous mode is used, the fairness is provided by conditions (14) and (15). In asynchronous CA, the property is consequence of the binomial probability distribution of cells chosen for the local operator application.

2.3. Operations on cellular arrays

When a phenomenon under simulation consists of several interacting processes, its CA-model should be composed of a number of CA which have to interact, executing some operations on intermediate results both on the local and on the global level. The problem in determining such an operation emerges when it turns to be incompatible with the alphabet of the CA-models under composition. For example, Boolean cellular arrays are incompatible with arithmetic addition. To solve this problem, a kind of *CA-composition algebra* on cellular arrays is introduced [28].

Like in any algebraic system, unary and binary operations are defined in the CA composition algebra.

Unary operators on cellular arrays. Two unary operators are defined: *averaging* which transforms Boolean cellular arrays into the equivalent real ones, and *state discretization*, which performs the inverse operation.

Averaging of the Boolean cellular array $\mathrm{Av}(\Omega_B)$ is a unary global operator which comprises the application of a local operator $\mathrm{Av}(m)$ to all cells of the cellular array, i.e., $\mathrm{Av}(\Omega_B) \in A_R \times M$, where $\Omega_B = \{(v,m) : v \in \{0,1\}, m \in M\}$, $\Omega_R = \{(u,m) : u \in [0,1], m \in M\}$.

The local operator $\mathrm{Av}(m)$ computes the average value of cell states in the *averaging area*

$$T_{\mathrm{Av}}(m) = \{m, \varphi_1(m), \ldots, \varphi_q(m)\}. \tag{18}$$

In the case of the 2D Cartesian cellular array, $T_{\mathrm{Av}}(i,j) = \{(i,j), (i+k, j+l) : k,l = -r, \ldots, r\}$, r being referred to as *averaging radius*.

Averaging may be regarded as a local operator

$$\mathrm{Av}(m) : S_{\mathrm{Av}}(m) \to (u,m), \quad u(m) = \langle v \rangle = \frac{1}{q}\sum_{k=0}^{q} v_k, \tag{19}$$

$S_{\mathrm{Av}}(m)$ having $T_{\mathrm{Av}}(m)$ as an underlying template. Angle brackets in (19) and in the sequel mean the averaged state values.

Discretization of a real cellular array $\mathrm{Dis}(\Omega_R)$ is a unary global operator

$$\mathrm{Dis}(\Omega_R) \in A_B \times M,$$

resulting from the application of a local operator $\mathrm{Dis}(m)$ to all cells of the cellular array. $\mathrm{Dis}(m)$ is a single-cell local operator that replaces a real state value $u \in [0,1]$ by 1 with probability $p = u$:

$$\mathrm{Dis}(m) : (u,m) \to (v,m), \quad v = \mathrm{Bool}(u) = \begin{cases} 1 & \text{if } u < \mathit{rand}, \\ 0 & \text{otherwise}, \end{cases} \tag{20}$$

where *rand* is a random number in the interval $[0,1]$, $\mathrm{Bool}(u)$ means a discretized value of $u \in [0,1]$. The above two unary operations are in the following relationship:

$$\begin{aligned} &\mathrm{Dis}(\Omega_B) = \Omega_B, \qquad &&\mathrm{Dis}(\mathrm{Av}(\Omega_B)) = \Omega_B, \\ &\mathrm{Av}(\Omega_R) = \Omega_R, \qquad &&\mathrm{Av}(\mathrm{Dis}(\Omega_R)) = \Omega_R. \end{aligned} \tag{21}$$

Binary operators on cellular arrays. Binary operators are defined on cellular arrays $\Omega \in A_B \times M_1 \cup A_R \times M_2$, if between $M_1 = \{(m_1)_i\}$ and $M_2 = \{(m_2)_i\}$ there exists an one-to-one correspondence $\xi : M_1 \to M_2$,

$$\begin{aligned} &(m_2)_i = \xi((m_1)_i), \qquad &&\forall (m_2)_i \in M_2, \\ &(m_1)_i = \xi^{-1}((m_2)_i), \qquad &&\forall (m_1)_i \in M_1. \end{aligned} \tag{22}$$

The cells $(v, ((m_1)_i)) \in \Omega_1$ and $(u, ((m_2)_i)) \in \Omega_2$ are further denoted as (v_i, m_1) and (u_i, m_2), respectively, which means that v_i and u_i are states in the corresponding cells of Ω_1 and Ω_2.

Binary operations are defined on the basis of the following principle: *ordinary arithmetic rules should be valid for the averaged forms of the operands*, i.e.,

$$\Omega_1 \diamondsuit \Omega_2 \Leftrightarrow \mathrm{Av}(\Omega_1) \diamond \mathrm{Av}(\Omega_2), \tag{23}$$

where $\diamondsuit$ stands for the cellular array addition $\oplus$, subtraction $\ominus$ or multiplication $\otimes$, and $\diamond$ stands for $+$, $-$, and $\times$, respectively.

Condition (23) may also be given in terms of ordinary arithmetics applied to cell states, i.e.

$$v_i(m_1) \diamond u_i(m_2) \Leftrightarrow \langle v_i(m_1) \rangle \diamond \langle u_i(m_2) \rangle \quad \forall i \in 1, \ldots, |M|, \tag{24}$$

where $\diamond$ stands for ordinary arithmetical $+$, $-$, and $\times$, respectively.

The reason for taking averaged state values as a generalized alphabet is twofold: 1) to allow ordinary arithmetic to be used for modeling spatial functions interactions, and 2) to make the results more comprehensive in terms of physics.

From (23) and (24) it follows that when all operands have real alphabets, the cellular array arithmetic coincides with the corresponding real cell-by-cell *arithmetical* rules. Otherwise the rules depend on the operands alphabets.

Let $\Omega_1 = \{(v, m_1) : v \in A_1, m_1 \in M_1\}$ and $\Omega_2 = \{(u, m_2) : u \in A_2, m_2 \in M_2\}$ be operands and $\Omega_3 = \{(w, m_3) : w \in A_3, m_3 \in M_3\}$ be a result, then binary operations are as follows.

The cellular array addition $\Omega_1 \oplus \Omega_2 = \Omega_3$. For different operands alphabets, the cellular addition looks somewhat different. The following cases are of major importance.

1. Both operands Ω_1 and Ω_2 are Boolean cellular arrays, Ω_3 is wanted to have a real alphabet, then according to (23) Ω_3 is computed as follows.

$$\begin{aligned} &\Omega_3 = \mathrm{Av}(\Omega_1) \oplus \mathrm{Av}(\Omega_2), \\ &w_i = \langle v_i \rangle + \langle u_i \rangle \quad \forall i = 1, \ldots, |M|. \end{aligned} \tag{25}$$

2. Both operands are Boolean and Ω_3 is also wanted to have a Boolean alphabet, then

$$\begin{aligned} &\Omega_3 = \mathrm{Dis}(\mathrm{Av}(\Omega_1) \oplus \mathrm{Av}(\Omega_2)), \\ &w_i = \begin{cases} 1 & \text{if } \mathit{rand} < (\langle u_i \rangle + \langle v_i \rangle), \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i = 1, \ldots, |M|. \end{aligned} \tag{26}$$

3. Both operands and the sum are Boolean, the latter being used as an intermediate result. So, it is convenient to update one of the operands, say Ω_2, so, that it be equal to the resulting array, i.e.,

$$\Omega_2(t+1) = \Omega_1(t) \oplus \Omega_2(t).$$

In that case, it suffices to invert a number of zero-states in the cells $(0, m_2) \in \Omega_2$. It should be done in such a way, that in every cell $(u_i, m_2) \in \Omega_2$ its averaged state value be increased by $\langle v_i \rangle$. According to (20), the probability of such an inversion is the relation of the value to be added to the number of "zeros" in the averaging area of each cell of Ω_2:

$$u_i' = \begin{cases} 1 & \text{if } u_i = 0 \ \&\ rand < \frac{\langle v_i \rangle}{1 - \langle u_i \rangle}, \\ u_i & \text{otherwise,} \end{cases} \qquad \forall i = 1, \ldots, |M|. \tag{27}$$

4. The operands have different alphabets. Let Ω_1 have a Boolean alphabet, Ω_2 has a real one, and Ω_3 is wanted to be a real cellular array, then

$$\begin{aligned} &\Omega_3 = \mathrm{Av}(\Omega_1) \oplus \Omega_2, \\ &w_i = \langle v_i \rangle + u_i \quad \forall i = 1, \ldots, |M|. \end{aligned} \tag{28}$$

5. Ω_1 has a Boolean alphabet, Ω_2 has a real one, and Ω_3 is wanted to be a Boolean real cellular array. Two ways are possible: 1) to discretize Ω_3, obtained by (28), and 2) to update Ω_1 using the following operation

$$w_i = \begin{cases} 1 & \text{if } v_i = 0 \ \&\ rand < \frac{u_i}{1 - \langle v_i \rangle}, \\ v_i & \text{otherwise,} \end{cases} \qquad \forall i = 1, \ldots, |M|. \tag{29}$$

Cellular array subtraction $\Omega_3 = \Omega_1 \ominus \Omega_2$. The following cases are worth to be considered.

1. Both operands are Boolean, the result is wanted to be real or Boolean. The operations are performed similar to those of the cellular addition. It is merely needed to replace "+" by "−" in (25) and (26).

2. Both operands are Boolean, and Ω_2 is to be updated to obtain $\Omega_2 = \Omega_1 \ominus \Omega_2$. In that case, some cell states $(1, m_2) \in \Omega_2$ should be inverted as follows:

$$u_i' = \begin{cases} 0 & \text{if } u_i = 1 \ \&\ rand < \frac{\langle v_i \rangle}{\langle u_i \rangle}, \\ u_i & \text{otherwise,} \end{cases} \qquad \forall i = 1, \ldots, |M|. \tag{30}$$

3. Ω_1 has a Boolean alphabet, Ω_2 has a real one, and Ω_3 is wanted to be a Boolean cellular array. Two ways are possible: 1) to discretize Ω_3, obtained by arithmetic subtraction, which performs the following operation:

$$\Omega_3 = \mathrm{Dis}(\mathrm{Av}(\Omega_1) - \Omega_2), \tag{31}$$

2) to update Ω_1 as follows

$$v'_i = \begin{cases} 0 & \text{if } u_i = 1 \ \& \ rand < \frac{v_i}{\langle u_i \rangle}, \\ u_i & \text{otherwise,} \end{cases} \qquad \forall i = 1, \ldots, |M|. \tag{32}$$

Cellular array multiplication $\mathbf{\Omega_3 = \Omega_1 \otimes \Omega_2}$. The operation is defined on real cellular arrays. The cell states are computed according to (25) with "$\times$" instead of "+". If any or both of the operands are Boolean, they should be averaged beforehand. Multiplication is used in the CA composition in those cases when one of the two operands is a constant cellular array, i.e., such one, where all cell states have the same value. This is helpful when subsets of cells have to be masked or scaled.

Since addition and subtraction are defined on cellular arrays with the alphabet restricted by the interval $[0, 1]$, the same condition should be satisfied for all cells in the resulting cellular arrays. If this is not so, the alphabet is to be renormalized.

Having a set of operations on cellular arrays in hands, it is possible to formulate the CA composition techniques. General composition principles prescribe to distinguish sequential, parallel, and intermixed composition techniques. The sequential composition represents several CA for processing one and the same cellular array by alternating their application at each iteration. The parallel composition suggests each CA to process its own cellular array, albeit having neighborhoods in the others. Both sequential and parallel types of composition have their versions for a local and a global levels of operation. The global composition techniques require that each CA be applied to a result of the global operator application. The local composition allows the application of local operators of different CA in any order during an iteration.

3. The sequential composition techniques

Sequential composition represents a common functioning of several CA, referred to as *components*. Their local operators are applied in a certain order to one and the same cellular array. This type of composition is conceptually identical to the CA *superposition*. It comprises a number of techniques differing in ordering of component operators application to the cellular array under processing.

Global superposition suggests the synchronous alternation of components global operators application. When those operators use different alphabets, their compatibility should be provided by transforming a Boolean cellular array into a real one, and vice versa. Apart from the general case of global superposition, two particular cases are worth to be distinguished: 1) self-superposition, which is in fact a multistage mode of a CA operation, and

2) so-called trivial sequential composition [22] which is the superposition of evolutions.

Local superposition is the composition when at each iteration the local operators of all components involved in the composition are applied in any order or at random. Naturally, the components should be asynchronous CA.

3.1. Global superposition

A number of CA form a global superposition $\aleph = \Psi_{Gl}(\aleph_1, \ldots, \aleph_n)$, $\aleph = \langle A_k, M, \hat{M}_k, \theta_k, \rho_k \rangle$ if its global operator $\Phi(\Omega)$ is the result of the sequential application of the global operators Φ_k to $\Omega_k = \Phi_{k-1}(\Omega_{k-1})$, $k = 1, \ldots, n$, providing compatibility of A_k and A_{k-1}, i.e.,

$$\Phi(\Omega) = \Phi'_n(\Phi'_{n-1}(\ldots \Phi'_1(\Omega_1))), \tag{33}$$

each Φ'_k being itself a superposition of Φ_k and a unary operator, i.e.

$$\Phi'_k = \Phi_k(\mathrm{Un}(\Omega_k)), \tag{34}$$

where

$$\mathrm{Un}(\Omega_k) = \begin{cases} \mathrm{Av}(\Omega_k) & \text{if } A_k = [0,1] \ \& \ A_{k-1} = \{0,1\}, \\ \mathrm{Dis}(\Omega_k) & \text{if } A_k = \{0,1\} \ \& \ A_{k-1} = [0,1]. \end{cases}$$

Components of the superposition may differ in alphabets, local operators and modes of operating, but the same naming set should be used.

The following particular cases of the global superposition are of special importance: self-superposition, trivial superposition, and the general type of superposition of CA with different types of alphabets.

Global Self-Superposition is the simplest superposition, being defined only for synchronous CA. A CA $\aleph = \langle A, M, \hat{M}, \theta, \sigma \rangle$ is a self-superposition $\aleph = \Psi_{\mathrm{Ss}}(\aleph_1, \ldots, \aleph_n)$, $\aleph_k = \langle A, M, \hat{M}_k, \theta, \sigma \rangle$ if its components differ only in active subsets $\hat{M}_k$.

Since the same local operator is applied at all stages of the superposition, there is no need to take care about their compatibility, so,

$$\Phi(\Omega) = \Phi_n(\Phi_{n-1}(\ldots(\Phi_1(\Omega)))).$$

The self-superposition is usually obtained by modifying a synchronous CA-model of a process which requires several neighboring cells to be updated simultaneously. Since it causes violation of non-contradictoriness condition (14), one should sacrifice some amount of cellular parallelism by dividing the iteration into a number of stages, which is done as follows:

1. Each k-th iteration is divided into n stages $t_1, \ldots, t_n$, the results of the k-th stage being $\Omega(t_k)$.

2. At the k-th stage, θ is applied to all cells named $m_k \in \hat{M}_k$ of $\Omega(t_k)$.

A bright manifestation of a synchronous self-superposition is a well-known two-stage CA-model of diffusion, which is described and studied in [1, 3, 10].

Example 1. Diffusion is a random wandering of particles aiming at the even distribution. It may be simulated by exchanging cell states in any pair of the adjacent cells. Since the synchronous simulation of such a process is contradictory, as is shown in Section 2.2, the self-superposition of two CA: $\aleph_1 = \langle A, M, \hat{M}_1, \theta, \sigma\rangle$ and $\aleph_2 = \langle A, M, \hat{M}_2, \theta, \sigma\rangle$ is used, where $A = \{0, 1\}$, $M = \{(i, j) : \ i, j = 0, 1, \ldots, N\}$,

$$\begin{aligned} \hat{M}_1 &= \{(i, j) : \ i \bmod 2 = 0, \ j \bmod 2 = 0\}, \\ \hat{M}_2 &= \{(i, j) : \ i \bmod 2 = 1, \ j \bmod 2 = 1\}, \end{aligned} \tag{35}$$

$\hat{M}_1$ and $\hat{M}_2$ being referred to as *even active subset* and *odd active subset*, respectively. The local operator is as follows:

$$\begin{aligned} \theta : \{(v_0, (i, j)), (v_1, (i, j + 1)), (v_2, (i, j + 1)), (v_3, (i, j + 1))\} \rightarrow \\ \{(u_0, (i, j)), (u_1, (i, j + 1)), (u_2, (i, j + 1)), (u_3, (i, j + 1))\}, \end{aligned} \tag{36}$$

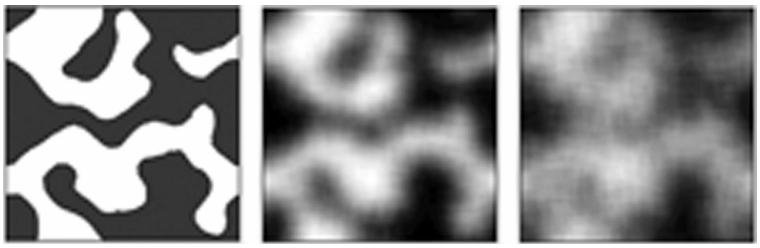
where

$$u_k = \begin{cases} v_{((k+1) \bmod 4)}, & \text{if } rand < p, \\ v_{((k-1) \bmod 4)}, & \text{if } rand > (1 - p), \end{cases} \qquad k = 0, 1, 2, 3,$$

the probability p depending on the diffusion coefficient.

Each iteration of a composed CA is divided into two stages: the even stage and the odd stage. At the odd stage, θ_1 is applied to all cells from $\hat{M}_1$; at the even stage, θ_2 is applied to all cells from $\hat{M}_2$.

In Figure 4, three snapshots are shown of the CA evolution simulating the diffusion of a black dye slopped onto the water surface.



$t = 0$ $t = 4$ $t = 8$

Figure 4. Snapshots of diffusion process, simulated by synchronous self-superposition of CA with a local operator (36). Black pixels stand for $v = 1$, white pixels — for $v = 0$

Global trivial superposition $\aleph = \Psi_{Tr}(\aleph_1, \ldots, \aleph_n)$, where $\aleph_k = \langle A_k, M, \hat{M}_k, \theta_k, \rho_k \rangle$, suggests the evolution of $\aleph$ is a sequential composition of the evolutions $\Sigma_{\aleph_k}(\Omega'_k(\hat{t}_k))$ of its components, where $\Omega'_k(\hat{t}_k)$ is a result of a unary operator(34) application to $\Omega_k(\hat{t}_k)$ if A_k and A_{k+1} are incompatible.

It is important to notice that the order of component application is essential.

Example 2. The pattern formation process starts in the cellular array obtained by a short-time application of a diffusion CA to a a cellular array with two areas of high concentration (black bands along the vertical borders) and empty (white) background (Figure 5(a)). Diffusion is simulated by an asynchronous probabilistic CA $\aleph_1 = \langle A, M, \hat{M}, \theta_1, \ \alpha \rangle$. The pattern formation is simulated by the synchronous CA $\aleph_2 = \langle A, M, \hat{M}\, \theta_2, \sigma \rangle$. Both CA have a Boolean alphabet $A = \{0, 1\}$, their naming sets are identical as well as active naming subsets, $M = \hat{M} = \{(i, j) : i, j = 0, \ldots, 300\}$.

Local operator of the diffusion CA $\aleph_1$ is as follows:

$$\begin{aligned} &\theta_1 : \{(v_0, (i,j)), (v_1, (i-1,j)), (v_2, (i,j+1)), (v_3, (i+1,j)), (v_4, (i-1,j))\} \\ &\rightarrow \ \{(u_0, (i,j)), (u_1, (i-1,j)), (u_2, (i,j+1)), (u_3, (i+1,j)), (u_4, (i-1,j))\} \end{aligned} \tag{37}$$

with the transition functions

$$\begin{aligned} &u_0 = v_k \quad \text{if } 0.25k < rand < 0.25(k+1), \\ &u_k = \begin{cases} v_0 & \text{if } 0.25k < rand < 0.25(k+1), \\ v_k & \text{otherwise,} \end{cases} \\ &k = 1, \ldots, 4. \end{aligned} \tag{38}$$

A local operator in the pattern formation CA $\aleph_2$ is as follows:

$$\theta_2 : (u_0, (i,j)) * \{(u_k, \phi_k(i,j)) : k = 1, \ldots, q\} \rightarrow (v_0, (i,j)), \tag{39}$$

where $q = (2r+1)^2$, $r = 3$ being a radius of the neighborhood template,

$$\phi_k(i,j) = (i + g_k, j + h_k), \quad g_k = (k \bmod (2r+1)) - r, \quad h_k = \lfloor k/(2r+1) \rfloor;$$

$$v_0 = \begin{cases} 1 & \text{if } \sum_{k=0}^{n} w_k v_k > 0; \\ 0 & \text{otherwise,} \end{cases} \tag{40}$$

where

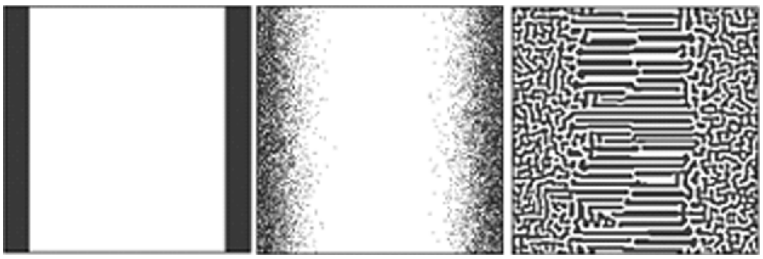
$$w_k = \begin{cases} 1 & \text{if } g \leq 1 \ \& \ h \leq 1 \\ -0.2 & \text{otherwise.} \end{cases}$$

The sum in (40) can be also obtained by imposing the weighted template

$$W = \begin{array}{|c|c|c|c|c|c|c|}\hline a & a & a & a & a & a & a \\ \hline a & a & a & a & a & a & a \\ \hline a & a & 1 & 1 & 1 & a & a \\ \hline a & a & 1 & 1 & 1 & a & a \\ \hline a & a & 1 & 1 & 1 & a & a \\ \hline a & a & a & a & a & a & a \\ \hline a & a & a & a & a & a & a \\ \hline \end{array}, \qquad a = -0.2,$$

onto a cell and computing the sum of products of its entries by underlying cell states.

In Figure 5, three snapshots of trivial composition of two CA ($\aleph_1$ simulating diffusion and $\aleph_2$ simulating the pattern formation) are shown. The cellular array size is 300×300, $\hat{t}_1 = 10$, $\hat{t}_2 = 12$. The pattern obtained is a stable one, further application of θ_2 to $\Omega_2(\hat{t}_2)$ implies no change in it.



a b c

Figure 5. Snapshots of the process, simulated by trivial superposition of asynchronous diffusion CA and synchronous pattern formation CA: a) initial array, b) $\hat{t}_1 = 10$, c) $\hat{t}_2 = 12$

Global superposition of arbitrary CA is a technique for obtaining a CA $\aleph_{Gl} = \Psi_{Gl}(\aleph_1, \ldots, \aleph_n)$, combining the operation of several CA $\aleph_k = \langle A_k, M, \hat{M}_k\theta_k, \rho\rangle$, $k = 1, \ldots, n$, whose alphabets and local operators are allowed to be incompatible, and modes of operation which may be different. The operation of the composed CA is as follows:

1. Each t-th iteration of the composed CA consists of n stages $t_1, \ldots, t_n$, the results of the t_k-th stage being $\Omega(t_k) = \Phi_{k-1}(t_{k-1})$.
2. At the k-th stage θ_k is applied to all cells $m \in \hat{M}_k$ of $\Omega'(t_k)$. The latter should be obtained by transformation of $\Omega(t_k)$ according to (34) if needed.

Example 3. Simulation of the alga spreading over the water is considered to combine three elementary processes:

1) agglomeration of randomly distributed alga,
2) diffusion of alga into water,

3) procreation of alga.

The first process is represented by a Boolean CA [31] $\aleph_1$ which is sometimes called a phase-separation CA [22], the second — by the two-stage diffusion CA $\aleph_2$ given in Example 1 (Section 3.1), the third — by $\aleph_3$ computing a nonlinear logistic function [32] in each cell. Accordingly, each iteration of the composed CA has three stages.

At the first stage t_1, the transition $\Omega_1 \to \Omega_2$ is performed by application of $\aleph_1 = \langle A_1, M, \hat{M}_1, \theta_1, \sigma\rangle$ with a single-cell updating local operator, where $A_1 = \{0,1\}$, $M = \{(i,j) : i,j = 0,\dots,N\}$, $\hat{M} = M$, σ stands for the synchronous mode.

$$\theta_1(i,j)) : \tilde{S}(i,j) \to \{(v',(i,j))\} \qquad \forall (i,j) \in M, \tag{41}$$

where

$$\tilde{S}(i,j) = \{(v_k, \phi_k(i,j)) : \ \phi_k(i,j) = (i+g, j+h), \ g,h \in \{-2,0,2\}\},$$

and

$$v' = \begin{cases} 1 & \text{if } s < 24 \text{ or } s = 25, \\ 0 & \text{if } s > 25 \text{ or } s = 24, \end{cases} \qquad \text{where} \quad s = \sum_{g=-2}^{2} \sum_{h=-2}^{2} v_{i+g,j+h}.$$

At the second stage, $\aleph_2$ performs a transition $\Omega_2 \to \Omega_3$ by application θ_2 (36) to all cells of Ω_2, the value of the probability in (36) being $p = 0.5$. As the alphabet A_2 is compatible with transition function (41), θ_2 is applied directly to the cells of Ω_2 resulting in a Boolean array $\Omega_3 = \{(u,(i,j))\}$.

At the third stage, the alga procreation CA $\aleph_3 = \langle A_3, M, \hat{M}_k, \theta_3, \sigma\rangle$ is applied to Ω_3. But since $A_3 = [0,1]$ and, hence, θ_3 is incompatible with Ω_3, the latter is transformed into Ω_3' by averaging: the operator $\mathrm{Av}(i,j)$ is applied to Ω_2 replacing each cell state $(v(i,j)$ by $\langle v(i,j)\rangle$ computed according to (19) with the template $T_{\mathrm{Av}}(i,j) = \{(i+k, j+l) : k,l = -8,\dots,8\}$. The local operator θ_3 is as follows:

$$\theta_3 : (\langle u(i,j)\rangle, (i,j)) \to (F(\langle u(i,j)\rangle), (i,j)), \tag{42}$$

which replaces the cell state $\langle u(i,j)\rangle$ by the value of a nonlinear logistic function

$$F(\langle u(i,j)\rangle) = 0.5\langle u(i,j)\rangle(1 - \langle u(i,j)\rangle).$$

The resulting cellular array having real states should be discretized according to (20) in order to become compatible with $\Omega(t+1)$, which is the initial cellular array for the first stage in the $(t+1)$th iteration.

The composition has been applied to an initial Boolean cellular array Ω with $v = 1$ randomly distributed so that $\langle v(i,j)\rangle \approx 0.5$ for all $(i,j) \in M$, the boundary conditions being periodic.

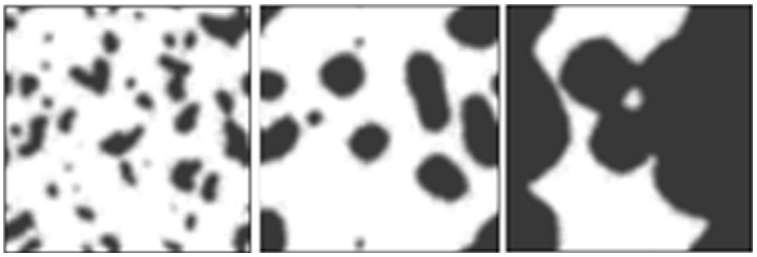


Figure 6. Snapshots of alga spreading in water, simulated by the synchronous global superposition of $\aleph_1$ with θ_1 (41), $\aleph_2$ with θ_2 (36) and $\aleph_3$ with θ_3 (42), black pixels stand for a maximal concentration of alga, white pixels — for clear water

In Figure 6, three snapshots of the simulation process are shown, the cellular arrays being averaged for making the observation more comprehensive. Black pixels stand for a maximum concentration of alga, white ones represent clear water. It is seen that in the first iterations, the total amount of alga decreases, but if some compact spots remain large enough, the procreation activity enhances their growth up to the saturation.

3.2. Local superposition

Asynchronous local superposition is mainly used in simulating biological processes and nano-kinetics, i.e., the processes on a micro- or nano- level, which according to their nature are considered to be completely stochastic. This technique aims at obtaining a CA-model $\aleph = \Psi_{Loc}(\aleph_1, \ldots, \aleph_n\}$ composed of asynchronous CA $\aleph_k = \langle A, M, \hat{M}_k, \theta_k, \alpha\rangle$, $k = 1, \ldots, n$, which differ only in local operators and (perhaps) in active subsets. The way of their common functioning is as follows. An iteration $\Omega(t) \to \Omega(t+1)$ consists of $|M|$ cycles, a cycle being a sequence of single-shot applications of $\theta_k(m)$, $k = 1, \ldots, n$, to a randomly chosen cell from $\Omega(t)$. Each θ_k is executed immediately after application. There is no constraints neither on the order of choosing a cell during an iteration, nor on the order of choosing θ_k for application during a cycle. Sometimes, the natural features of the process under simulation dictate a certain ordering or a grouping of local operators in a cycle, but usually they are randomly chosen.

Example 4. A chemical reaction of CO oxidation over platinum catalysts, well known in the surface chemistry as Ziff–Guilari–Barshod model [30], is represented by a local superposition of four simple local operators, mimicking an elementary action of adsorption, reaction, oxidation, and diffusion.

The cellular array Ω corresponds to a catalysts plate, each site on it being named as $(i, j) \in M$, $|M| = N \times N$, $\hat{M} = M$. The alphabet contains three symbols $A = \{a, b, 0\}$, so that $(a, (i, j)), (b, (i, j)$, and $(0, (i, j))$ are

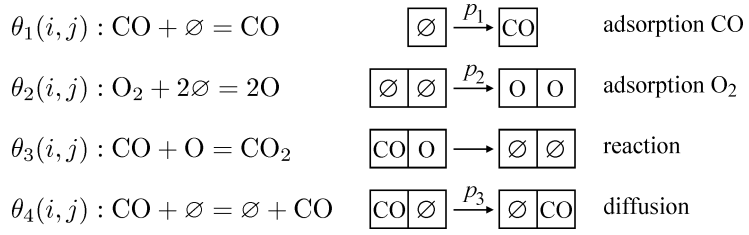


Figure 7. Graphical representation of local operators involved in an asynchronous local superposition simulating chemical oxidation of CO on platinum

cells corresponding to the sites occupied by the molecules of CO, O, or being empty, respectively. In the initial array, all cells are empty. The CO oxidation process consists of the following elementary molecular actions in any cell named (i, j) (Figure 7):

1) Adsorption of CO from gas: if the cell (i, j) is empty, it becomes occupied by a CO molecule with probability p_1, whose value depends on the partial pressure of CO in the gas above the plate.

2) Adsorption of oxygen O_2 from gas: if the cell (i, j) is empty and has an empty adjacent cell, both become occupied by an atom of oxygen with probability p_2. The probability depends on the partial pressure of oxygen in the gas. One out of $h < 4$ adjacent cells of the cell (i, j) is chosen with probability $p_n = 1/h$.

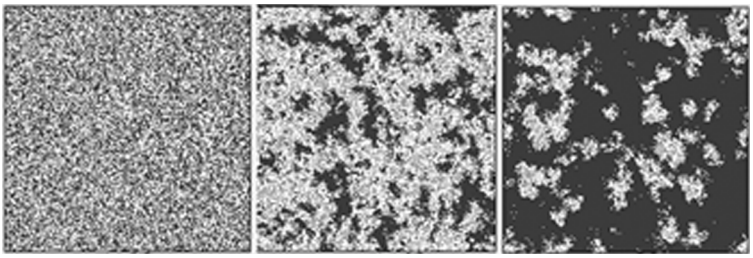
3) Reaction of oxidation of CO ($CO + O \to CO_2$): if the cell (i, j) occurs to be in a CO state and its adjacent cell is in O state, then the molecule CO_2, formed by reaction, transits to the gas and both cells become empty. One out of $h < 4$ adjacent cells occupied by oxygen is chosen with probability $p_n = 1/h$.

4) Diffusion of CO over the plate: if the cell (i, j) occurs to be in a CO state when one of its adjacent cells is empty, the cell (i, j) becomes empty, and the empty cell gets the state CO. This occurs with probability p_3. One out of $h < 4$ adjacent cells of the cell (i, j) is chosen with probability $p_n = 1/h$.

Formally, local operators of the above actions are represented as follows:

$$\begin{aligned}
\theta_1(i,j):&\ \{(0,(i,j))\} \to \{(a,(i,j))\} \quad \text{if } p_1 > rand,\\
\theta_2(i,j):&\ \{(0,(i,j))(0,\phi_k(i,j))\} \to \{(b,(i,j)),(b,\phi_k(i,j))\}\\
&\ \text{if } (k-1)p_n < rand < kp_n \ \&\ p_2 > rand,\\
\theta_3(i,j):&\ \{(a,(i,j))(b,\phi_k(i,j))\} \to \{(0,(i,j)),(0,\phi_k(i,j))\}\\
&\ \text{if } (k-1)p_n < rand < kp_n,\\
\theta_4(i,j):&\ \{(a,(i,j))(0,\phi_k(i,j))\} \to \{(0,(i,j)),(a,\phi_k(i,j))\}\\
&\ \text{if } (k-1)p_n < rand < kp_n) \ \&\ p_3 > rand,
\end{aligned} \tag{43}$$

for $k = 1, \ldots, 4$.



$t = 2$ $t = 40$ $t = 80$

Figure 8. Snapshots of the oxidation reaction simulation by an asynchronous superposition of local operators (43). Black pixels stand for CO, gray pixels — for O, and white pixels — for empty sites

In Figure 8, three snapshots of the simulation process are shown, the initial cellular array $\Omega(0) = \{(0,(i,j)) : \forall (i,j) \in M\}$, $|M| = 200 \times 200$.

In the general case, the local superposition is neither commutative nor associative, i.e., if $\theta_1 \neq \theta_2 \neq \theta_3$, then

$$\theta_1(\theta_2(m)) \neq \theta_2(\theta_1(m)), \qquad \theta_3(\theta_2(\theta_1(m)) \neq (\theta_3(\Phi_2))(\theta_1(m)). \tag{44}$$

The above two properties are very important, because the results of the simulation may essentially differ if the order of superpositions is changed. Although in case of a long evolution, a repetitive sequence of superpositions, for example, such as $\theta_1(\theta_2(\theta_1(\theta_2(m)\ldots)))$, makes the composition insensitive to the substitution being the first. If it is not the case, the only way to make the result independent of the order of substitutions in the composition is their random choice at any step of application (Monte Carlo method).

4. Parallel CA composition

The parallel composition suggests functioning of a number of n interacting CA, each processing its own cellular array. Taking into account the fact that the number of possible interactions in the composition exponentially increases with n, and for clearness of presentation, the composition $\aleph = \Upsilon(\aleph_1, \aleph_2)$ of not more than two CA is further considered. The components $\aleph_k = \langle A_k, M_k, \hat{M}_k, \theta_k, \rho_k \rangle, k = 1, 2$, are allowed to have different alphabets, different modes of operation, different local operators, and between $M_1 = \{(m_1)_i\}$, and $M_2 = \{(m_2)_i\}$, $i = 1, 2 \ldots, |M|$, condition (22) is satisfied.

Since θ_1 and θ_2 are to be executed simultaneously, the computation is dangerous from the point of view of non-contradictoryness condition (14), which states that no local operator may attempt to update simultaneously one and the same cell. On the other hand, in order that the component transition functions in θ_1 and θ_2 interact, one should use the same cell state variables. Hence, with respect to (11) and (13), the left-hand sides of θ_1 and θ_2 should have a non-empty intersection, i.e.,

$$(S_1((m_1)_i) \cup S_1''((m_1)_i) \cap (S_2((m_2)_i) \cup S_2''((m_2)_i) \neq \emptyset.$$

Combining this statement with (14), the condition for a parallel composition yields

$$T_k((m_i)_k) \subseteq M_k, \tag{45}$$

$$T_k''((m_i)_k) \subseteq (M_1 \cup M_2) \quad \forall k \in \{1, 2\}. \tag{46}$$

From (45) it follows that $\theta_1((m_1)_i)$ and $\theta_2((m_2)_i)$ may update cells only from their own cellular array, from (46) they are allowed to use cell states of the both. This means, that the neighborhoods of the cells $(m_1)_i$ and $(m_2)_i$, may intersect only by their contexts.

The above conditions are valid both for local and global composition techniques, as well as both for CA with synchronous and asynchronous modes of operation.

4.1. Global parallel composition

Similar to a sequential case, a trivial parallel CA composition is also distinguished.

A trivial parallel composition $\aleph = \Upsilon_{Tr}(\aleph_1, \aleph_2)$, $\aleph_k = \langle A_k, M_k, \hat{M}_k, \theta_k, \rho_k \rangle$, $k = 1, 2$, is a degenerate particular case of parallel composition, when the components are completely independent, i.e., their neighborhood templates do not intersect at all, i.e.

$$(S_1((m_1)_i) \cup S_1''((m_1)_i) \cap (S_2((m_2)_i) \cup S_2''((m_2)_i) = \emptyset. \tag{47}$$

Nonetheless, after both components have terminated, a binary operation on the resulting cellular arrays may be performed. Hence, the result of the composed CA computation is

$$\Omega(\hat{t}) = \Omega_1(\hat{t}_1) \diamondsuit \Omega_2(\hat{t}_2), \tag{48}$$

where $\diamondsuit$ is any binary operator given in Section 2.3.

Example 5. Two phase separation models are to be compared by computing a difference between two resulting cellular arrays:

1) $\Omega_1(\hat{t}_1)$ obtained by the evolution of a totalistic CA $\aleph_1 = \langle A_1, M_1, \hat{M}_1, \sigma \rangle$, which is already described in Example 3 (Section 3.1) with θ_1, given as (41), and

2) $\Omega_2(\hat{t}_2)$ obtained by solving a PDE proposed in [33], which describes the same process,

$$\frac{\partial u}{\partial t'} = 0.2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - 0.2(u-1)(u-0.5)(u-0.9). \tag{49}$$

The finite-difference representation of (49) is a synchronous CA $\aleph_2 = \langle A_2, M_2, \hat{M}_2, \theta_2, \sigma\rangle$, where $A_2 = [0,1]$, $M_2 = \hat{M}_2 = \{(i,j)_2 : i = x/h, j = y/h, i,j = 0,\ldots,N\}$, h being a space step, $t = t'/(\Delta t)$ — iteration time,

$$\theta_2 : (u_0,(i,j)_2) * \{(u_1,(i-1,j)_2),(u_2,(i,j+1)_2),(u_3,(i+1,j)_2),(u_4,i,j-1)_2)\} \\ \rightarrow (u'_0,(i,j)_2), \tag{50}$$

where

$$u'_0 = \frac{u_1 + u_2 + u_3 + u_4 - 4u_0}{2h}.$$

The initial cellular arrays $\Omega_1(0)$ and $\Omega_2(0)$ for $\aleph_1$ and $\aleph_2$ are identical, having equal distribution of "ones" and "zeros", so, that $\langle v(i,j)_1\rangle = u(i,j)_2 = 0.5$ for all $(i,j)_1 \in M_1$ and all $(i,j)_2 \in M_2$.

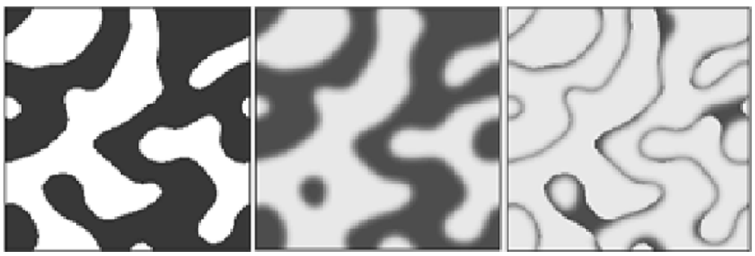
The comparison of the results of both components

$$\Omega_1(\hat{t}_1) = \{(v,(i,j)_1) : \ v \in A_1, \ (i,j)_1 \in M_1\},$$
$$\Omega_2(\hat{t}_2) = \{(u,(i,j)_2) : \ u \in A_2, \ (i,j)_2 \in M_2\},$$

is done by computing the absolute value of their cellular subtraction defined in Section 2.3. Since $\Omega_1(\hat{t}_1)$ and $\Omega_2(\hat{t}_2)$ are incompatible the first is to be averaged according to (19). The final result is obtained as $\Omega'_2(\hat{t}) = \{(u'_2,(i,j)_2)\}$, where

$$u'_2((i,j)_2) = |\langle v_1((i,j)_1)\rangle - u((i,j)_2)|. \tag{51}$$

The three resulting cellular arrays: $\Omega_1(\hat{t}_1)$, $\Omega_2(\hat{t}_2)$, and $\Omega'_2(\hat{t})$ are shown in Figure 9.



a b c

Figure 9. Snapshots of a parallel trivial composition of two CA simulating the phase separation: a) resulting cellular array obtained by a totalistic CA (41), b) resulting cellular array obtained by a CA based on PDE (50), and c) their difference. Black pixels stand for 1, white for 0, the gray-scale intensity corresponds to values from $[0,1]$

Nontrivial parallel composition $\aleph = \Psi(\aleph_1, \aleph_2)$ suggests that all component $\aleph_k$, $k = 1, 2$, interact at each iteration. Two types of interaction between $\aleph_1$ and $\aleph_2$ determine two types of parallel composition techniques: unidirectional parallel composition and bidirectional parallel composition [22].

In *unidirectional parallel composition*, one of the components, say $\aleph_1$, evolves independently, i.e., the transition functions of θ_1 do not depend on the states of cells from Ω_2. But, the transition functions of $\aleph_2$ depend of states of cells from both cellular arrays. Hence, conditions (45, 46) take the following form

$$T_1''((m_1)_i) \subseteq M_1, \qquad \forall (m_1)_i \in M_1, \tag{52}$$

$$T_2''((m_2)_i) \subseteq (M_1 \cup M_2) \quad \forall (m_2)_i \in M_2. \tag{53}$$

Such a kind of composition is frequently used when simulating a certain process by $\aleph_1$, an auxiliary CA, let it be $\aleph_2$, is added for transforming simulation results of $\aleph_1$ into a proper form for analyzing or visualizing its evolution. For example, $\aleph_1$ is a Boolean CA, and observation of its evolution requires it to be real values. Then, $\aleph_1$ evolves independently, and $\aleph_2$ performs the averaging of $\Omega_1(t)$ at each iteration using cell states of Ω_1 in its transition functions.

In *bidirectional parallel composition*, the transition functions of both components depend on the states of cells from both cellular arrays, i.e.,

$$\begin{aligned} &T_1''((m_1)_i) \subseteq (M_1 \cup M_2) \quad \forall (m_1)_i \in M_1, \\ &T_2''((m_2)_i) \subseteq (M_1 \cup M_2) \quad \forall (m_2)_i \in M_2, \end{aligned} \tag{54}$$

(45) being preserved as well. If the alphabets of $\aleph_1$ and $\aleph_2$ are incompatible, then a suitable unary transformation should be done after each iteration on $\Omega_1(t)$ and $\Omega_2(t)$.

Example 6. A 2D reaction–diffusion process of propagation of an autocatalytic reaction is simulated by bidirectional parallel composition of two CA:

1) a two-stage synchronous diffusion CA $\aleph_1 = \langle A_1, M_1, \hat{M}_1, \theta_1, \sigma \rangle$ is given in Example 1 (Section 3.1), and
2) a single cell synchronous CA $\aleph_2 = \langle A_2, M_2, \hat{M}_2, \theta_2, \sigma \rangle$, which computes a real nonlinear function of the cell state.

Since A_1 and A_2 are incompatible, unary operators $\mathrm{Av}(i, j)$ and $\mathrm{Dis}(i, j)$ are to be added to the local operators θ_1 and θ_2, respectively. This may be done as follows:

$$\begin{aligned} \theta_1 : &\{(v_0, (i,j)_1), (v_1, (i, j+1)_1), (v_2, (i+1, j)_1), (v_3, (i, j-1)_1)\} * \\ &\{(u_0, (i,j)_2), (u_1, (i, j+1)_2), (u_2, (i+1, j)_2), (u_3, (i, j-1)_2)\} \to \\ &\{(v_0', (i,j)_1), (v_1', (i, j+1)_1), (v_2', (i+1, j)_1), (v_3', (i, j-1)_1)\}, \end{aligned} \tag{55}$$

where

$$v'_k = \begin{cases} \text{Bool}(u_{(k+1) \bmod 4}) & \text{if } rand < p, \\ \text{Bool}(u_{(k-1) \bmod 4}) & \text{if } rand > (1-p). \end{cases}$$

In reaction CA $\aleph_2$, the local operator $\theta_2((i,j)_2)$ is combined with $\text{Av}((i,j)_1)$, which results in the following:

$$\theta_2 : (u,(i,j)_2) * \{S_{\text{Av}}((i,j)_1) \rightarrow \{f(\langle v((i,j)_1)\rangle,(i,j)_2)\} \tag{56}$$

where

$$f(\langle v((i,j)_1)\rangle) = 0.5\langle v((i,j)_1)\rangle(1-\langle v((i,j)_1)\rangle),$$

$\langle v(i,j)_1)\rangle$ being obtained according to (19).

The simulation space is a 300×300 cells square area with some rectangular obstacles through which the reaction cannot penetrate (in Figure 10, shown in light gray). The initial condition is a little spot of a reactant of high density in the central part of the top side of the area, which corresponds to the cellular array with an aggregate of "ones" in-between the obstacles at the top. The front propagation is shown in Figure 10 by six snapshots of the simulation process.

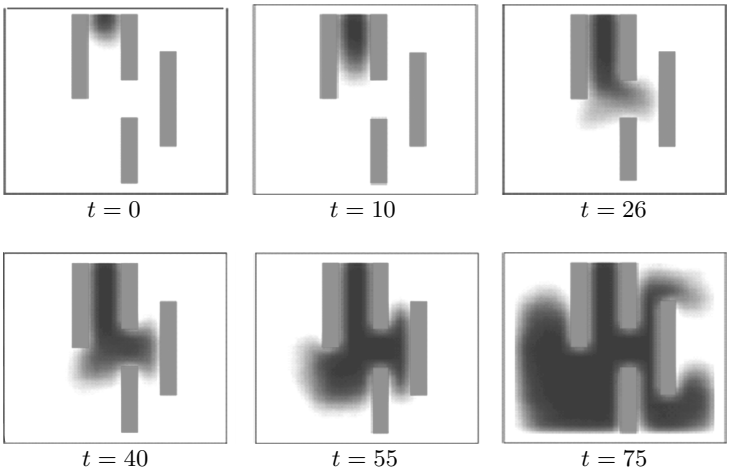


Figure 10. Snapshots of $\aleph_2$ evolution of a parallel bidirectional composition simulating the front propagation of autocatalytic reaction. Gray pixels stand for obstacles, black pixels — for a maximal concentration of the reactant, white — for the absence of a reactant

4.2. Local parallel composition

Like in the sequential case, this type of a composition is aimed at obtaining an asynchronous CA-model $\aleph = \Upsilon_{\text{Loc}}(\aleph_1, \aleph_2)$ composed of two asynchronous CA $\aleph_k = \{A_k, M_k, \hat{M}_k, \theta_k, \alpha\}$, $k = 1,2$. The components may differ in

alphabets and in local operators, naming sets M_1 and M_2 being in relation (22). The way of the composed CA functioning is as follows.

Both components operate in parallel in asynchronous mode: at each tth iteration, θ_k is applied to all cells of $\hat{M}_k$, the cells being selected in any order and updated immediately after selection. All variations of the cell selection ordering are allowed, namely, the following ones are possible:

1) random selection of cells in both CA independently according to given probability distribution, usually the binomial distribution is used;
2) random selection of cells but one and the same for both CA, the cells $(m_i)_1$ and $(m_i)_2 = \xi(m_i)_1$ being updated simultaneously;
3) prescribed order of cell selection, one and the same in both CA;
4) alternation of θ_1 and θ_2 application with any order of cells selection.

Like in the global parallel composition case, the local parallel composition may be unidirectional and bidirectional, depending on the number (one or both) of components having local configuration contexts in the cellular array of the other component.

Example 7. A soliton-like 1D process is simulated by a parity totalistic CA [4] $\aleph_1 = \{A_1, M_1, \hat{M}_1, \theta_1, \alpha\}$. Since A_1 is a Boolean alphabet the process is difficult to recognize as two moving waves passing one through the other. So, to make the process observable in a habitual form, $\aleph_1$ is composed with another CA $\aleph_2 = \{A_2, M_2, \hat{M}_2, \theta_2, \alpha\}$ which performs averaging of any cell state in Ω_1 just after its updating. Thus, the composition of $\aleph_1$ and $\aleph_2$ is local unidirectional: $\aleph_1$ operates independent of $\aleph_2$, and $\aleph_2$ uses cell states of Ω_1 as the context in θ_2. The naming sets $M_1 = \hat{M}_1 = \{i_1 : i = 0, \ldots, N\}$, and $M_2 = \hat{M}_2 = \{i_2 : i = 0, \ldots, N\}$, are in one-to-one correspondence, i.e., $i_2 = \xi((i_2))$. A local operator

$$\theta_1 : (v_0, i_1) * \{(v_j, i_1 + j) : \ j = -r, \ldots, -1, 1, \ldots, r\} \to (v_0', i_1), \qquad (57)$$

where

$$v_0'(i_1) = \begin{cases} 1 & \text{if} \quad w \neq 0 \ \& \ w = 0 \bmod 2, \\ 0 & \text{otherwise,} \end{cases} \qquad w = \sum_{j=-r}^{r} v_j. \qquad (58)$$

The mode of $\aleph_1$ operation is an ordered asynchronous one: θ_1 is applied sequentially to $i_1 = 0, 1, \ldots N$, the cells (v, i_1) being immediately updated. Hence, in transition function (58), the cell states $v(i_1 + j)$ with $j < 0$, which are leftwards of i_1, are already in the next state, while the rightward cells are still in the current state. The boundary conditions are periodic. The initial global cellular state $\Omega_1(0)$ has certain patterns referred to as "particles" [4]. Here, the two following particles are used: $P_1 = 1101$, and $P_2 = 10001001$

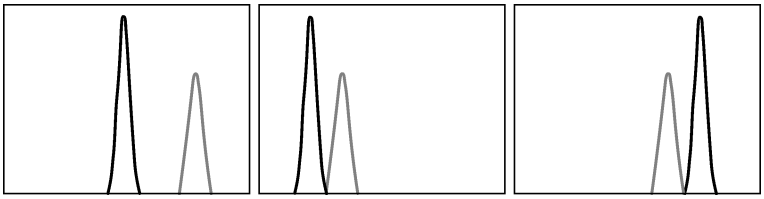


Figure 11. Snapshots of the soliton propagation obtained by simulating the process using a local parallel composition of two CA with local operators given by (57) and (59)

with $r = 4$. All other cells are in zero states. The evolution of $\aleph_1$ shows that the first particle P_1 appears in $\Omega_1(t)$ any two iterations being displaced by $d_1 = 7$ cells to the left. And the second particle P_2 appears in $\Omega_1(t)$, any 6 iteration being displaced by $d_2 = 12$ cells also to the left. So, while each six iterations P_1 are displaced by 21 cells, and P_2 is displaced by 12 cells. Hence, a distance between particles diminishes by nine cells in six iterations. After the start ($t = 0$) during the period from $t = 12$ up to $t = 24$, particles are superimposed, and after $t = 30$ the first particle is ahead, as is shown in the following global states:

$t = 0$: 0000 . . . 0000000000000**1000100**1000000000000000000000000**1101**100
$t = 6$: 0000 . . . 00**1000100**1000000000000000**1101**1000000000000000000000000
$t = 30$: 000000000000000000000000**1101**10000**1000100**100 . . . 0000000000000
$t = 36$: 00**1101**1000000000000000**1000100**100000 . . . 0000000000000000000

The second CA $\aleph_2$ performs an asynchronous averaging of Ω_1, in order to transform the patterns displacement into the waves propagation. The steps of $\aleph_2$ are synchronized with those of $\aleph_1$, and the order of cell selection is the same:

$$\theta_2 : (v_0, i_1) * \{(v_j, i_1 + j) : \ j = -r, \ldots, -1, 1, \ldots, r\} \rightarrow (\langle v_0 \rangle, i_1), \qquad (59)$$

where

$$\langle v_0 \rangle = \frac{1}{(2r+1)} \sum_{j=-r}^{r} v_j.$$

4.3. The mixed composition

In practice, the simulation of complex phenomena requires a number of CA to be included in different type of the composition forming a complicated scheme of composition techniques. The main principle for constructing such a mixed composition is that any component may represent a composed CA. Hence, the mixed composition is a hierarchical structure, any level of hierarchy being a composed CA.

Example 8. A simplified process of vapor nucleation in a binary system (vapor, gas–carrier) is simulated using a mixed CA composition. The process has been studied in a number of investigations on self-organizing reaction-diffusion systems. For example, in [34], an attempt is made to solve the PDE system which describes the process as follows:

$$\begin{aligned}\frac{\partial v}{\partial t} &= 0.025\Big(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\Big)+0.2v-v^3-1.5u,\\ \frac{\partial u}{\partial t} &= 0.0025\Big(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\Big)+v-u.\end{aligned} \tag{60}$$

Since two species are involved in the process, a bidirectional parallel composition should be used. The resulting CA $\aleph = \Upsilon(\aleph_1, \aleph_2)$ has two components, each simulating a reaction-diffusion process in $\Omega_1 = \{(v,(ij)_1)\}$ and $\Omega_2 = \{(u,(ij)_2)\}$, respectively. The component $\aleph_k = \Psi_{\mathrm{Gl}}(\aleph_{Dk}, \aleph_{Rk})$, in its turn, is a sequential composition of $\aleph_{Dk} = \langle A_D, M_k, \hat{M}_k, \theta_{Dk}, \beta\rangle$ which represents the diffusion, and $\aleph_{Rk} = \langle A_R, M_k, \hat{M}_k, \theta_{Rk}, \sigma\rangle$, which represents the reaction. The two diffusions CA, $\aleph_{D1}$ and $\aleph_{D2}$, independently operate, each in its own cellular array. Their results are used by the reaction CA $\aleph_{R1}$ or $\aleph_{R2}$, which are in the bidirectional parallel composition with each other. Since the alphabets A_D and A_R are incompatible, the diffusion global operator result $\Phi_{Dk}(\Omega_{D_k}(t))$ is averaged and that of the reaction $\Phi_{Rk}(\Omega_{R_k}(t))$ is discretized, which yields the following superposition of global operations of $\aleph_k$:

$$\Phi_k(\Omega_k(t)) = \mathrm{Dis}(\Phi_{Rk}(\mathrm{Av}(\Phi_{Dk}(\Omega_k(t-1))))), \quad k=1,2. \tag{61}$$

Diffusion is simulated by the two-stage synchronous CA given in Example 1 (Section 3.1) with θ_{Dk} given as (36). The difference between $\aleph_{D1}$ and $\aleph_{D2}$ is in the values of probabilities used in the transition function. They are taken as $p_v = 0.5$, $p_u = 0.05$, which corresponds to the diffusion coefficients in (60) with the time step $\Delta t = 0.6$ s and the space step $h = 0.1$ cm.

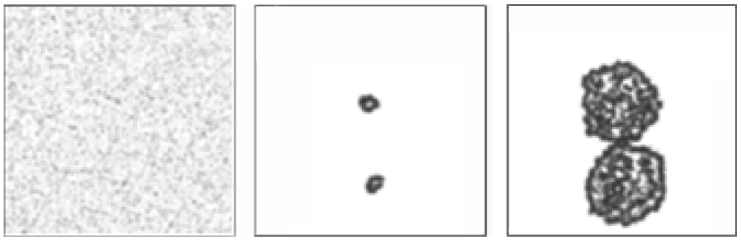
Reaction is simulated by a single cell context-free CA with the following local operators:

$$\begin{aligned}&\theta_{R1} : (\langle v\rangle, (i,j)_1) \to (f_v(\langle v((i,j)_1)\rangle, \langle u((i,j)_2)\rangle), (i,j)_1),\\ &\theta_{R2} : (\langle u\rangle, (i,j)_2) \to (f_u(\langle v((i,j)_1)\rangle, \langle u((i,j)_2)\rangle), (i,j)_2),\end{aligned} \tag{62}$$

where

$$f_v(x,y) = 0.2x - x^3 - 1.5y, \qquad f_u(x,y) = x - y.$$

The size of both cellular arrays is 300×300 cells with periodic boundary conditions. The initial conditions are Boolean cellular arrays with the following evenly distributed concentrations of vapor and gas: $\langle v(i,j)_1\rangle = 0.1$, $\langle v(i,j)_2\rangle = 0.9$.



$\Omega_1(0)$ $\Omega_1(12)$ $\Omega_1(40)$

Figure 12. Snapshots of the vapor nucleation process obtained by simulating it as a parallel composition of two superpositions or diffusion and reaction. Black pixels stand for the vapor particles

The evolutions of $\aleph_1$ and $\aleph_2$ show the processes of vapor and gas distribution in time, respectively. In Figure 12, three snapshots are shown for the vapor nucleation process.

5. Computational properties of composed CA

In real simulation tasks, when dealing with a large CA size and a large number of iterations, the computational properties, such as accuracy, stability, and complexity are of main importance. Hence, the impact of the above composition techniques on these properties should be assessed. As for the accuracy, the study of this property is focused on the procedures which are beyond the conventional cellular automata theory, i.e., the cellular array operations, which may contribute some errors. The stability assessment of the composition directly depends on the same property of its components, which may exhibit different kinds of behavior [2], their evolutions tending to a stable state or never reaching it, or being chaotic. So, the attention is focused on the stability conservation, provided the components of CA composition are stable. The property of complexity is concerned with additional operations, which are incorporated to eliminate incompatibility between the interacting components.

It should be noted that contemporary mathematics has no well-established concepts of CA computational properties, as well as no methods for their quantitative assessment. So, the subsections below may be regarded as some considerations for the problem, indicating to the points for further investigation.

5.1. Accuracy of the composed CA

One of Boolean CA advantages is that they are absolutely accurate from the computational standpoint, i.e., there are no rounding off errors. But, once the averaging $\mathrm{Av}(\Omega)$ or the discretization $\mathrm{Dis}(\Omega)$ is used and, hence,

real numbers are processed, the errors may be brought in.

In trivial compositions, both sequential and parallel, the two above operations are performed only once at the beginning and at the end of the simulation process, bringing in inessential approximation error. But in non-trivial compositions, when $\mathrm{Av}(\Omega)$ and $\mathrm{Dis}(\Omega)$ are used in each iteration, their impact on the result may be significant. So, just this pair of operations are further considered from the point of view of the accuracy problem.

Let Ω_B be the tth iteration result of a composed CA, and $\Omega_R = \mathrm{Av}(\Omega_B)$ should be obtained to make the next operation compatible. Then according to (19) the Boolean states $(v, m) \in \Omega_B$ are replaced by real ones from the finite set of numbers $Q = \{0, 1/q, \ldots, 1\}$, where $q = |\mathrm{Av}(m)|$. Hence, the error $E_{\mathrm{Av}}(m)$ incorporated by approximating a Boolean representation of a spatial function by discrete values from a finite set Q is constrained by

$$E_{\mathrm{Av}} \leq \frac{1}{|\mathrm{Av}(m)|} = \frac{1}{q}. \tag{63}$$

A Boolean discretization of $\Omega_R = \{(u, m)\}$ performed according to (20) and resulting in $\Omega_B = \{(v, m)\}$ also brings in some errors. Probabilistic formula (20) provides that the obtained Ω_B in its averaged form is equal to $Av(\Omega_B)$, which means that the following equalities condition the accuracy of discretization:

$$\Omega_R = \mathrm{Av}(\Omega_B), \quad u(m) = \langle v(m) \rangle \quad \forall m \in M, \tag{64}$$

discretization error $E_{\mathrm{Dis}}(m)$ being the difference

$$E_{\mathrm{Dis}}(m) = |u(m) - \langle v(m) \rangle|. \tag{65}$$

The error vanishes in those cells where

$$u(m) = \langle v(m) \rangle = \frac{1}{q} \sum_{k=0}^{q-1} v(\phi_k(m)), \tag{66}$$

which happens very rarely, for example, when a fragments of a linear function or a parabola of odd degree is discretized. The error is most serious in the cells, where $u(m)$ has extremes.

The most correct representation of the discretization error is a function $E_{\mathrm{Dis}}(m, t)$, which shows possible deviations of $u(m, t)$ in all cells during the evolution. But sometimes in particular cases, error values in a certain part of M, or a maximal error in extremes of the spatial function at a certain time is of interest. For a general assessment of the CA composition, the mean discretization error at a given $t = \hat{t}$

$$E_{\mathrm{Dis}}(\hat{t}) = \frac{1}{|M|} \sum_{m \in M} |u(m, \hat{t}) - \langle v(m, \hat{t}, \rangle|, \tag{67}$$

is also used.

From (66) and (67) it follows that discretization errors depend on the averaging area size q and on the smoothness of $u(m)$ on $T_{\mathrm{Av}}(m)$. Both these parameters are conditioned by the discretization step h, which should be taken small, allowing q to be chosen large enough to smooth the extremes. Since $h = \mathbf{S}/|M|$, where $\mathbf{S}$ is a physical space under simulation, a small h means a large cellular array size.

The following experiment gives the quantitative insight into the accuracy problem.

Example 9. A half-wave of a sinusoid $u = \sin x$, $0 < x < \pi$, is chosen for the experimental assessment of the discretization error dependence of via $|M|$ and $\mathrm{Av}(m)$. The cellular array representation of a given continuous function is as follows:

$$\Omega = \Big\{(u(m), m) : \ u(m) = \sin \frac{\pi m}{|M|}, \ m = 0, 1, \ldots, |M|\Big\}. \tag{68}$$

To obtain the dependence of $E_{\mathrm{Dis}}(|M|)$, 30 discretizations $\{\mathrm{Dis}_k(\Omega) : k = 1, 2, \ldots, 30\}$ of (68) have been obtained with $|M_k| = 60 \times k$, which corresponds to the argument domain $60 < |M_k| < 1800$, or in the angular form $2^\circ > h > 0.1^\circ$. Each $\mathrm{Dis}_k(\Omega)$ has been averaged with $|\,\mathrm{Av}_k(m)| = 0.2|M_k|$, and the mean errors $E_{\mathrm{Dis}}(|M_k|)$ have been computed according to (67) (Figure 13).

To obtain the dependence $E_{\mathrm{Dis}}(q)$, $q = |\,\mathrm{Av}(m)|$, 30 discretizations $\{\mathrm{Dis}_j(\Omega) : j = 1, 2, \ldots, 30\}$ of (68) have been obtained with fixed $|M| = 360$ but different $q_j = |\,\mathrm{Av}_j\,|$, where $q_j = 5 \times j$. Each $\mathrm{Dis}_j(\Omega)$ has been averaged with $\mathrm{Av}_j(m)$, and the mean errors $E_{\mathrm{Dis}}(q_j)$ have been computed according to (67) (Figure 14).

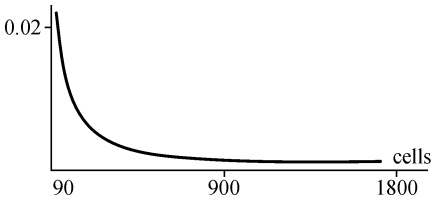


Figure 13. The mean discretization error dependence on the naming set size $|M|$ with $|\,\mathrm{Av}(m)| = 0.2|M|$ for cellular array (68)

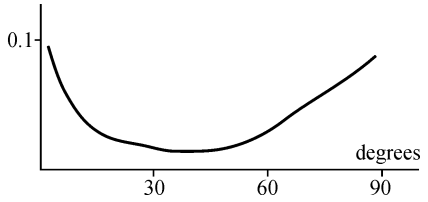


Figure 14. The mean discretization error dependence on the size of the averaging area $|\,\mathrm{Av}(m)|$ with $|M| = 360$ for cellular array (68)

From Figures 13 and 14 we may be conclude the following:

1) the mean error $E_{\mathrm{Dis}}(|m|)$ follows the increase of $|M|$ and does not exceed 1 % with $|M| > 360$, corresponding to $h < 0.5°$;
2) the mean error $E_2(|\,\mathrm{Av}(m)|)$ has a minimum when $|\,\mathrm{Av}(m)| \approx 36°$, i.e. $q_{E_{\mathrm{Dis}}=\min} = 0.2|M|$.

From this example it follows that regulating the smoothness of the extremes by an appropriate choice of the CA size the needed accuracy of the CA composition may be attained. Of course, the complexity of simulation increases with an increase of $|M|$.

5.2. The CA composition stability

There are two aspects of stability concerning the CA composition. The first is *behavioral stability*, which determines whether the CA evolution tends to a stable state or to a periodic cycling. The property is studied for simple Boolean CA in [2], but no method is known to check behavioral stability for an arbitrary CA. As for CA with real alphabets and nonlinear functions, their behavioral stability is a subject of the nonlinear dynamic system theory and may be checked using its methods, as is usually done in the continuous mathematics (see, for example, [35]). The problem which is of interest in connection with CA composition is as follows: all CA component being stable, is the composition obtained by the above techniques also stable? Till now the problem remains open. Moreover, the current level of knowledge about the CA behavior stability does not seem to be high enough for its solution.

The second stability aspect is *computational stability*. This property is associated with the round-off errors, which are inevitable when the floating point arithmetics is used. This aspect of stability is more effectual for investigation because there are at least two particular cases of composition methods, for which the computational stability may be quantitatively assessed.

The first case comprises the local and the global sequential compositions of Boolean CA-models. Since all alphabets are Boolean, there are no round-off errors, and since cellular arrays under processing have a finite size, the resulting averaged values are bounded and stable.

The second case includes sequential or parallel global composition techniques of Boolean and real CA-models, where cellular array transformations $\mathrm{Av}(\Omega_R)$ and $\mathrm{Dis}(\Omega_B)$ are used at each iteration. In this case, the following assertion is true: if $\aleph_R$ is stable, and, hence, its state values can be bounded by the real closed interval $[0, 1]$, then the composition is computationally stable. This assertion is evident at the same time it is of considerable importance for widely used diffusion–reaction processes, because it states that

composition of Boolean diffusion and real reaction is free of the so-called Courant constraint imposed on the PDE counterpart of the process. The courant constraint in the PDE explicit solution is associated with second order partial derivatives of spatial coordinates (Laplace operator), representing a diffusive part in the PDE. It forbids, for example for the 2D case, the value $C_{\rm PDE} = \frac{\tau d}{h^2}$ to exceed 0.25, where h is a spatial step, d is a diffusion coefficient. From the above follows that the time step $\tau < \frac{1}{2}\frac{h^2}{d}$, should be small enough, which results in a significant increase of computation time. Meanwhile, the diffusion part may be simulated by a CA without any constraint. For example, a CA given in Example 1 (Subsection 3.1), has the Courant constant $C_{\rm CA} = 1.5$, which is proved in [10]. Another example of CA-diffusion is proposed in [26], where $C_{\rm CA}$ depends on an averaging radius as follows: $C_{\rm CA} = r(r+1)/6$ for 2D case.

Example 10. A diffusion–reaction process called a propagating front is simulated by two models: (1) an explicit finite-difference method of the PDE solution and (2) composition of a Boolean diffusion CA and a real reaction CA. The PDE is as follows:

$$\frac{\partial u}{\partial t} = \frac{\partial u^2}{\partial x^2} + \frac{\partial u^2}{\partial y^2} + 0.5u(1-u), \tag{69}$$

where $d = 0.33$ cm^2/s. The discretized 2D space is $M = \{(i,j) : i,j = 0,1,\ldots,639\}$. The initial state for the PDE solution is

$$u^{(0)}(i,j) = \begin{cases} 1 & \text{if } 280 < i,j < 360, \\ 0 & \text{otherwise.} \end{cases}$$

The finite difference representation of (69)

$$u^{(t+1)}(i,j) = u^{(t)}(i,j) + 0.33\big(u^{(t)}(i-1,j) + u^{(t)}(i+1,j) + u^{(t)}(i,j+1) + u^{(t)}(i,j-1) - 4u^{(t)}(i,j)\big). \tag{70}$$

With the time-step $\tau = 1$ s and the spatial step $h = 1$ cm, the Courant value $C_{\rm PDE} = \tau d/h^2 = 0.33$, which is out of the Courant constraint. So, the function $u^{(t)}(i,j)$ obtained with (70) is not stable.

The same process may be simulated by the superposition $\aleph = \Psi(\aleph_{\rm diff}, \aleph_{\rm reac})$. The first component $\aleph_{\rm diff} = \langle A, M, \hat{M}, \theta_1, \sigma\rangle$ is an approximation of a Laplace operator by averaging as it is proposed in [26], where $A = [0,1]$, $q = |Av_r(i,j)|$, $M = \{(i,j) :\ i,j = 0,1,\ldots,639\}$, $\hat{M} = M_1$,

$$\theta_1 : S_{\rm Av}(i,j) \to (u,(i,j)), \quad u(i,j) = \frac{1}{q}\sum_{{\rm Av}(i,j)} u(\phi_k(i,j)). \tag{71}$$

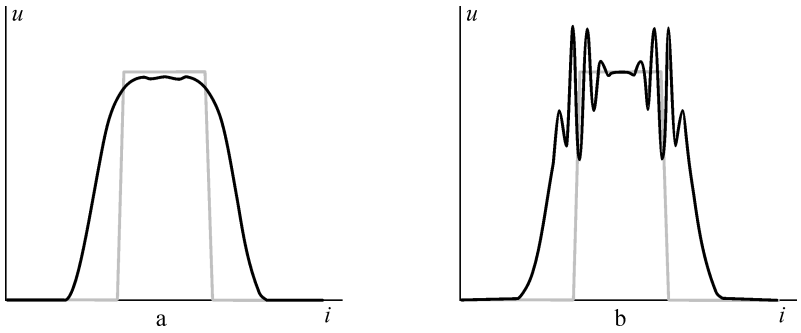


Figure 15. Simulation of a 2D propagation front initiated by a dense square in the central part of a cellular space, the profile $u(i, 319)$ is obtained for $t = 20$ by: a) CA superposition of $\aleph_{\text{diff}}$ with θ_1 (71) and $\aleph_{\text{reac}}$ with θ_2 (72), b) solution to finite-difference equation (70)

The second component $\aleph_{\text{reac}} = \langle A, M, \hat{M}, \theta_2, \sigma \rangle$ is a single-cell CA computation in each cell a reaction function $f = 0.5u(1-u)$ with subsequent discretization, where $A, M, \hat{M}, \sigma$ are equal to those of $\aleph_1$, the local operator being

$$\theta_2 : (u, (i, j)) \to (v, (i, j)), \quad v = \text{Dis}(0.5u(1-u)). \tag{72}$$

Snapshots of both processes (PDE solution) and (CA superposition) after 20 iterations are shown in Figure 15. It is seen that evolution of CA superposition is absolutely stable, while a finite difference solution to (71) exhibits a divergence.

5.3. Composition complexity

Here, an attempt is made to assess how much of additional work a composed CA has to do as compared to the total complexity of the components. Such an assessment cannot be precisely done. There are many reasons for that. The most significant are the following: (1) complexity relations between different arithmetic operations strongly depend on hardware architecture; (2) the same is true for comparing Boolean and real operations; (3) complexity of performing CA transition functions range from $O(n)$ to $O(2^n)$, n being the cardinality of the neighborhood in the local operator. Nonetheless, an insight may be given into the relation between the complexity of transition function computation and that of transformations needed for providing compatibility.

In case when the sequential local asynchronous composition and global synchronous composition techniques contain no averaging and discretization operations, no additional time is needed, and the total number of elementary operations is equal to the sum of those of the component CA.

When the global synchronous composition, no matter sequential or parallel, is used, transformation of a Boolean cellular array into a real one and vice versa is to be performed at each iteration. In such a case, the iteration time is increased by the following number of additional elementary operations.

$$t_{\text{add}} = (t_{\text{Av}} + t_{\text{Dis}})|M|, \tag{73}$$

where t_{Av} and t_{Dis} are numbers of elementary operations which have to be executed by a cell while performing the averaging according to (19), or discretization according to (20), respectively. As for t_{Av}, it is clearly seen from (19) that the time needed to compute $\text{Av}(v_m)$ may be assessed as $t_{\text{Av}} = \tau C_{\text{Av}} |\text{Av}(m)|$ where $C_{\text{Av}} \approx 1$ is a constant, τ being the time of elementary function execution; $t_{\text{Dis}} = \tau C_{\text{rand}}$ according to (20) depends only on a random number generator used, which may be taken $C_{\text{rand}} < 5$. Since the transformation is used in the composition techniques where both Boolean and real components are included, the time t_{add} should be compared to the transition functions computation time $t_{\text{comp}} = t_B + t_R$, where $t_B = \tau C_B$ and $t_R = \tau C_R$. The coefficients C_B and C_R essentially depend on the character of the transition functions but, usually, both functions require to execute not more than 100 elementary operations.

Comparison of t_{add} with t_{comp} yields:

$$\frac{t_{\text{add}}}{t_{\text{comp}}} = \frac{C_{\text{Av}} + C_{\text{Dis}}}{C_B + C_R},$$

which enables us to conclude that t_{add} and t_{comp} have identical order of complexity, hence, Boolean–real transformations twice increase the computation time.

6. Conclusion

Till now, no mathematical method and no promising approach are known to the CA synthesis from a given description of its evolution. Nevertheless, some way out should be found. A simple one is to follow a well-known approach used in the PDE theory which implies composing PDE systems out of a set of differential operators and functions. Such an approach seems to be expedient when considering the following similarities between the CA composition and the PDE system construction. For example, first order and second order differential operators in PDEs with respect to space have their CA counterparts in the form of shift and diffusion local operators, respectively. And in both cases for obtaining a mathematical model of the reaction–diffusion process, those operators are composed with nonlinear reaction functions. Unfortunately, the above similarities are only particular cases. In general, there is no formal procedure to obtain a CA simulating

the space-time nonlinear behavior. It is just the fact that has provoked the development of compatible algebraic operations on cellular arrays, allowing one to integrate continuous functions into a CA composition techniques.

But the most important destination of the CA composition is not in presenting another way of simulating processes which may be described in terms of PDE, but in obtaining the capability of constructing mathematical models for those phenomena for whom no other mathematical description is known. Such processes are mostly associated with the fields of science which are in the initial stage of development. For example, plant growth, embryo fetation, cellular division, morphogenesis (biology); surface oxidation, chemical reaction on catalyst, dissociation, adsorption (chemistry); epitaxial growth, crack formation, rubber deformation, robotics (engineering); tumor growth (medicine), etc. Of course, the available experience in science and engineering is not sufficient to forecast the future of the CA simulation methodology. Anyway, now it is clear that only a small part of the huge amount of CA have evolutions which resemble natural phenomena, and, hence, may be used for simulation. Moreover, those, which occur to be helpful, ought to be enriched with some additional properties, such as probability into transition functions, complicated modes of operations, a composite alphabet, a non-homogeneous cellular space, etc. All these being oriented to obtain CA-models of complex phenomena require a unique formalism for composing complex CA-models from a number of simpler ones. The above considerations allow us to hope that the presented attempt to construct a systematic approach to the CA composition is not futile.

References

[1] Toffolli T. Cellular Automata as an alternative to (rather than approximation of) differential equations in modeling physics // Physica D. — 1984. — Vol. 10. — P. 117–127.

[2] Wolfram S. A New Kind of science. — USA Champaign III: Wolfram Media Inc., 2002.

[3] Bandman O. Comparative Study of Cellular Automata Diffusion Models // PaCT-1999 / V. Malyshkin, ed. — Berlin: Springer, 1999. — P. 395–404. — (LNCS; 1662).

[4] Park J.K., Steiglitz K., Thurston W.P. Soliton-like behavior in automata // Physica D. — 1986. — Vol. 19. — P. 423–432.

[5] Vannozzi C., Fiorentino D., D'Amore M., Rumshitzki D.S., Dress A., Mauri R. Cellular Automata model of phase transition in binary mixtures // Indust. Eng. Chem. Res. — 2006. — Vol. 45, No. 4. — P. 2892–2896.

[6] Creutz M. Celllular Automata and self-organized criticality // Some new directions in science on computers / G. Bannot, P. Seiden, eds. — Singapore: World Scientific, 1996. — P. 147–169.

[7] Rothman D.H., Zalesky S. Lattice-Gas Cellular Atomata. Simple Model of Complex Hydrodynamics. – UK: Cambridge University Press, 1997.

[8] Succi S. The Lattice Boltzmann Equation for Fluid Dynamics and Beyond. — NY: Oxford University Press, 2001.

[9] Axner L., Hoekstra A.G., Sloot P.M.A. Simulating Time Harmonic Flows with the Lattice Boltzmann Method. — 2007. — (Phys. Rev.; E-75, 036709+7).

[10] Malinetski G.G., Stepantsov M.E. Modeling diffusive processes by Cellular Automata with Margolus neighborhood // Zhurnal Vychislitelnoy Matematiki i Mathematicheskoy Phisiki. — 1998. — Vol. 36, No. 6. — P. 1017–1021.

[11] Frish U., d'Humieres D., Hasslacher B., Lallemand P., Pomeau Y., Rivet J.P. Lattice-gas hydrodynamics in two and three dimensions // Complex Systems. — 1987. — No. 1. — P. 649–707.

[12] Elokhin V.I., Latkin E.I. Matveev A.V. Gorodetskii V.V. Application of statistical lattice models to the analysis of oscillatory and autowave processes in the reaction of carbon monoxide oxidation over platinum and palladium surfaces // Kinetics and Catalysis. — 2003. — Vol. 4, No. 5. — P. 672–700.

[13] Jansen A.P.J. An Introduction to Monte-Carlo Simulation of Surface Reactions. — Vol. 1, 3. — 2003. — (arXiv; cond-mat/0303028).

[14] Neizvestny I.G., Shwartz N.L., Yanovitskaya Z.Sh., Zverev A.V. 3D-model of epitaxial growth on porous {111} and {100} Si Surfaces // Comp. Phys. Comm. — 2002. — Vol. 147. — P. 272–275.

[15] Saum M.A., Gavrilets S. CA Simulation of Biological Evolution in Genetic Hyperspace // ACRI-2006 / S. El Yacoubi, B. Chopard, S. Bandini, eds. — Berlin: Springer, 2006. — P. 3–13. — (LNCS; 4176).

[16] Wu Y., Chen N., Rissler M., Jiang Y., Kaiser D., Alber M. CA Models of Myxobacteria Sworming // ACRI-2006 / S. El Yacoubi, B. Chopard, S. Bandini, eds. — Berlin: Springer, 2006. — P. 192–203. — (LNCS; 4176).

[17] Ghaemi M., Shahrokhi A. Combination of the Cellular Potts Model and Lattice Gas Cellular Automata for Simulating the Avascular Cancer Growth // ACRI-2006 / S. El Yacoubi, B. Chopard, S. Bandini, eds. — Berlin: Springer, 2006. — P. 297–303. — (LNCS; 4176).

[18] Slimi R., El Yacoubi S. Spreadable Probabilistic Cellular Automata Models // ACRI-2006 / S. El Yacoubi, B. Chopard, S. Bandini, eds. — Berlin: Springer, 2006. — P. 330–336. — (LNCS; 4176).

[19] Bandini S., Manzoni S., Vizzari G. SCA: a model to simulate crowding dynamics // IEICE Trans on Inf. Syst. — 2004. — Vol. E, 87. — P. 669–676.

[20] Adamatzky A. Dynamics of Crowd-minds. Pattern of Irrationality in Emotions, Beliefs and Actions. — World Scientific, 2005.

[21] Biondini F., Bontempi F., Frangopol D.M., Malerba P.G. // J. Struct Eng. — 2004. — 130 (11). — P. 1724–1737.

[22] Bandman O. Composing Fine-Grained Parallel Algorithms for Spatial Dynamics Simulation // PaCT-2005 / V. Malyshkin, ed. — Berlin: Springer, 2005. — P. 99–113. — (LNCS; 3606).

[23] Wolfram S. Universality and Complexity in Cellular Automata // Physica D. — Amsterdam, 1984. — Vol. 10. — P. 1–35.

[24] Chua L.O. CNN: a Paradigm for Complexity. — Singapore: World Scientific, 2002.

[25] Achasova S., Bandman O., Markova V., Piskunov S. Parallel Substitution Algorithm. Theory and Application. — Singapore: World Scientific, 1994.

[26] Weimar L.R., Boon J.-P. Class of Cellular Automata for reaction-diffusion systems // Phys. Rev. — 1994. — Vol. E, 49. — P. 1749–1752.

[27] Bandman O. Simulating Spatial Dynamics by Probabilistic Cellular Automata // ACRI-2002 / S. Bandini, B. Chopard, M. Tomassini, eds. — Berlin: Springer, 2002. — P. 10–20. — (LNCS; 2493).

[28] Bandman O. Spatial functions approximation by Boolean arrays // Bull. Novosibirsk Comp. Center. Ser. Computer Science. — Novosibirsk, 2003. — Iss. 19. — P. 10–19.

[29] Bandman O. Coarse-Grained Parallelization of Cellular-Automata Simulation Algorithms // PaCT-2007 / V. Malyshkin, ed. — Berlin: Springer, 2007. — P. 370–384. — (LNCS; 4671).

[30] Ziff R.M., Gulari E., Barshad Y. Kinetic phase transitions in an irreversible surface-reaction model // Phys. Rev. Lett. — 1986. — Vol. 56. — P. 2553.

[31] Vichniac G. Simulating physics by Cellular Automata // Physica D. — 1984. — Vol. 1. — P. 86–112.

[32] Svirezhev Yu. Nonlinear Waves, Dissipative Structures and Catastrophes in Ecology. — Moscow: Nauka, 1987.

[33] Schlogl F. Chemical reaction models for non-equilibrium phase transitions // Zh. Physik. — 1972. — Vol. 253. — P. 147–161.

[34] Schrenk C.P., Schutz P., Bode M., Purwins H. Interaction of Selforganised quaziparticles in two-dimensional Reaction Diffusion System: the Formation of molecules // Phys. Rev. – 1998. — Vol. E, 5, No. 6. — P. 6481–5486.

[35] Michel A.N., Wang K., Hu.B. Qualitative Theory of Dynamics Systems: The Role of Stability Preserving. — NY: CRC Press, 2001.